EVIDENTLY AI

LLM-as-a-judge: using LLMs for evaluations

A complete guide

June 2025

Contents

About this guide

LLM-as-a-judge is a common technique to evaluate LLM-powered products.

It grew popular for a reason: it's a practical alternative to costly human evaluation when assessing open-ended text outputs.

Judging generated texts is tricky, whether it's a "simple" summary or a chatbot conversation. Metrics like accuracy don't work well because there are many ways to be "right" without exactly matching the example answer. And things like style or tone are subjective and hard to pin down.

Humans can handle these nuances, but manually reviewing every response doesn't scale. LLM-as-a-judge emerged as an alternative: you can use LLMs to evaluate the generated texts. Interestingly, the LLM is both the source of the problem and the solution!

In this guide, we'll cover:

- How LLM-as-a-judge works and why it's effective.
- Types of LLM judges for both offline and online evaluations.
- How to build an LLM evaluator and craft good prompts.
- Pros, cons, and alternatives to LLM evaluations.

While there's plenty of great research on this topic — we'll reference some of it — we're going to keep things practical. This guide is for anyone working on an LLM-powered product and wondering if this technique could work for them.

TL;DR

- **LLM-as-a-Judge** is an evaluation method to assess the quality of text outputs from any LLM-powered product, including chatbots, Q&A systems, or agents.
- It uses a large language model (LLM) with an **evaluation prompt** to rate generated text based on criteria you define.
- LLM judges can handle both **pairwise comparisons** (comparing two outputs) and **direct scoring** (evaluating output properties like correctness or relevance).
- It's not a single metric but a flexible technique for **approximating human judgment**. LLM judge evaluations are specific to your application.
- The success of its use depends on the prompt, model, and complexity of the task.
- LLM judges can be used for both **offline** and **online** evaluations.
- Pros are flexibility, cost-effectiveness, speed, and accessibility to domain experts.

What is LLM as a judge?

TL;DR: LLM-as-a-Judge uses LLMs to evaluate AI-generated texts based on custom criteria defined in an evaluation prompt.

As you build your LLM-powered product — whether it's a chatbot, code generator, or email assistant — you need to evaluate its quality.

- During development, to compare models or prompts and ensure you're improving.
- Once it's live, to monitor user interactions for quality and safety.
- Anytime you make changes, to run regression tests and ensure nothing breaks.

LLM-as-a-judge is an evaluation approach that supports all these workflows. The idea is simple: ask an LLM to "judge" the text outputs using guidelines you define.

Say, you have a chatbot. You can ask an external LLM to evaluate its responses, similar to how a human evaluator would, looking at things like:

- **Politeness**: Is the response respectful and considerate?
- **Bias**: Does the response show prejudice towards a particular group?
- **Tone**: Is the tone formal, friendly, or conversational?
- **Sentiment**: is the emotion expressed in the text positive, negative or neutral?
- **Hallucinations**: Does this response stick to the provided context?

To apply the method, you take the text output from your AI system and feed it back into the LLM, this time alongside an **evaluation prompt**. The LLM will then return a score, label, or even a descriptive judgment — following your instructions.

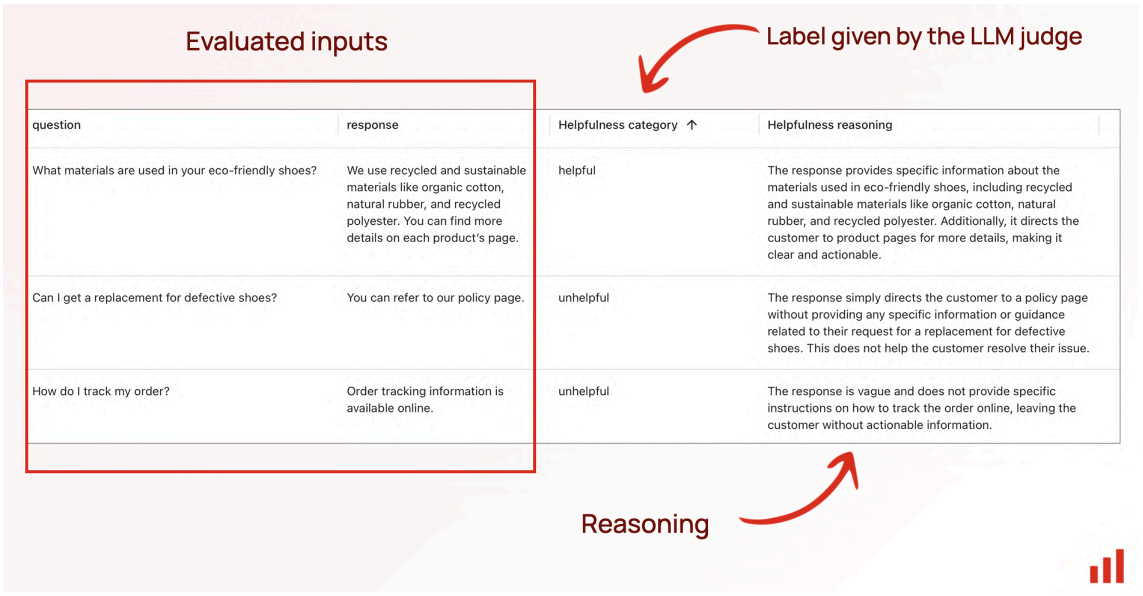
The beauty of this approach is that it lets you evaluate text outputs **automatically** and look at **custom properties** specific to your use case.

For example, you can instruct the LLM to judge the helpfulness of the customer chatbot's response.

Simplified prompt: *"Given a QUESTION and RESPONSE, evaluate if the response is helpful. Helpful responses are clear, relevant, and actionable. Unhelpful responses are vague, off-topic, or lacking detail. Return a label: helpful or unhelpful."*

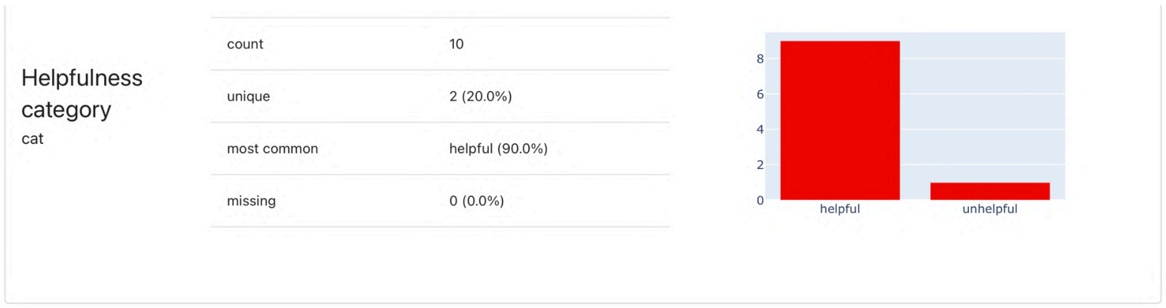
Side note: all prompt examples in this guide are simplified. The goal is to illustrate the idea.

If you run it over your chatbot outputs, the LLM judge will score each response.



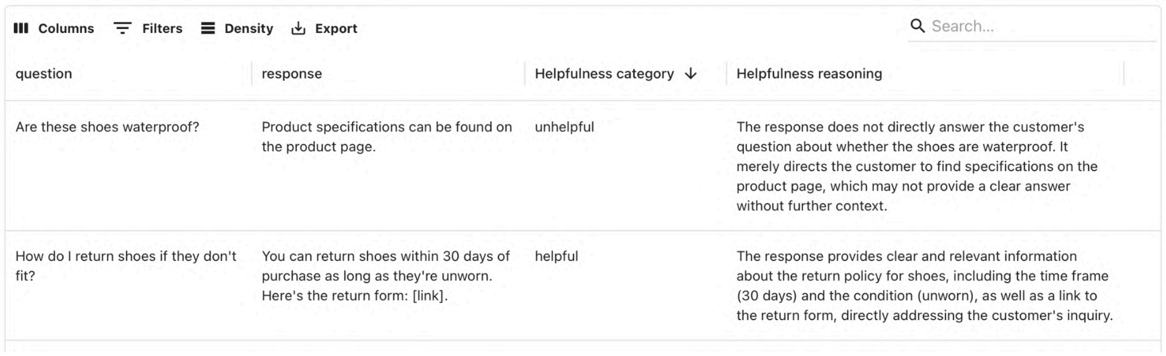
question	response	Helpfulness category ↑	Helpfulness reasoning
What materials are used in your eco-friendly shoes?	We use recycled and sustainable materials like organic cotton, natural rubber, and recycled polyester. You can find more details on each product's page.	helpful	The response provides specific information about the materials used in eco-friendly shoes, including recycled and sustainable materials like organic cotton, natural rubber, and recycled polyester. Additionally, it directs the customer to product pages for more details, making it clear and actionable.
Can I get a replacement for defective shoes?	You can refer to our policy page.	unhelpful	The response simply directs the customer to a policy page without providing any specific information or guidance related to their request for a replacement for defective shoes. This does not help the customer resolve their issue.
How do I track my order?	Order tracking information is available online.	unhelpful	The response is vague and does not provide specific instructions on how to track the order online, leaving the customer without actionable information.

You can then summarize the results over the entire conversation dataset to see the distribution: what is the share of helpful responses?



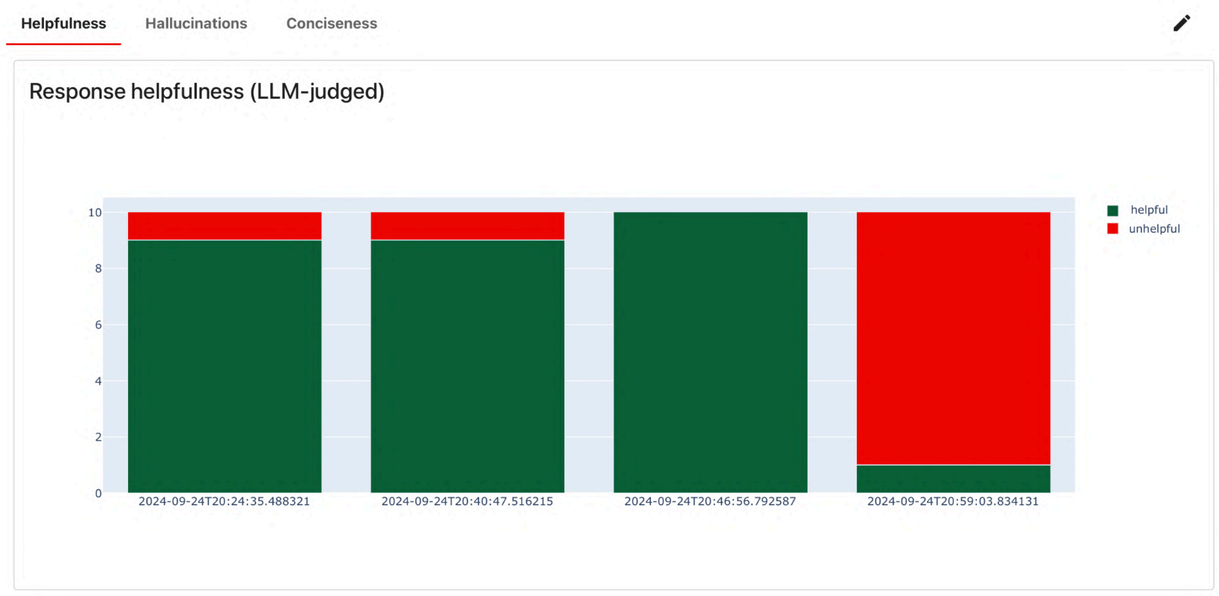
Helpfulness category
cat

count	10
unique	2 (20.0%)
most common	helpful (90.0%)
missing	0 (0.0%)



question	response	Helpfulness category	Helpfulness reasoning
Are these shoes waterproof?	Product specifications can be found on the product page.	unhelpful	The response does not directly answer the customer's question about whether the shoes are waterproof. It merely directs the customer to find specifications on the product page, which may not provide a clear answer without further context.
How do I return shoes if they don't fit?	You can return shoes within 30 days of purchase as long as they're unworn. Here's the return form: [link].	helpful	The response provides clear and relevant information about the return policy for shoes, including the time frame (30 days) and the condition (unworn), as well as a link to the return form, directly addressing the customer's inquiry.

It's not just about one-off analysis: you can run these evaluations on live data. This will let you track how well the chatbot works and detect issues, like a sudden surge in "unhelpful" responses.



Want to assess another property? Just write a new prompt!

It's worth noting that the LLM-as-a-judge **is not an evaluation metric** in the same sense as accuracy, precision, or NDCG. In machine learning, a metric is a well-defined, objective measure: they precisely quantify how well a model's predictions match the ground truth.

In contrast, LLM-as-a-judge **is a general technique** where you use LLM to approximate human labeling. When you ask an LLM to assess qualities like "faithfulness to source," "correctness," or "helpfulness," you define what these terms mean in the evaluation prompt and rely on the semantic relationships the LLM learned from training data.

The LLM follows your instructions, just like a human would. You can then track how many responses are labeled as "good," but this isn't a fixed, deterministic measure. It's a use-case-specific proxy metric.

The success of using LLM judges also heavily depends on the implementation details—the model you use, the prompt design, and the task complexity. You will also need to adapt the evaluation prompt to the specific evaluator LLM: both words and formats matter.

Before we dive into details, let's address the elephant in the room: how come you use LLMs to evaluate "its own" work? Isn't this cheating?

Why does it work?

TL;DR: Classifying text to assess its properties is easier than generating it. The evaluator is external to the main product and performs a simpler, more focused task.

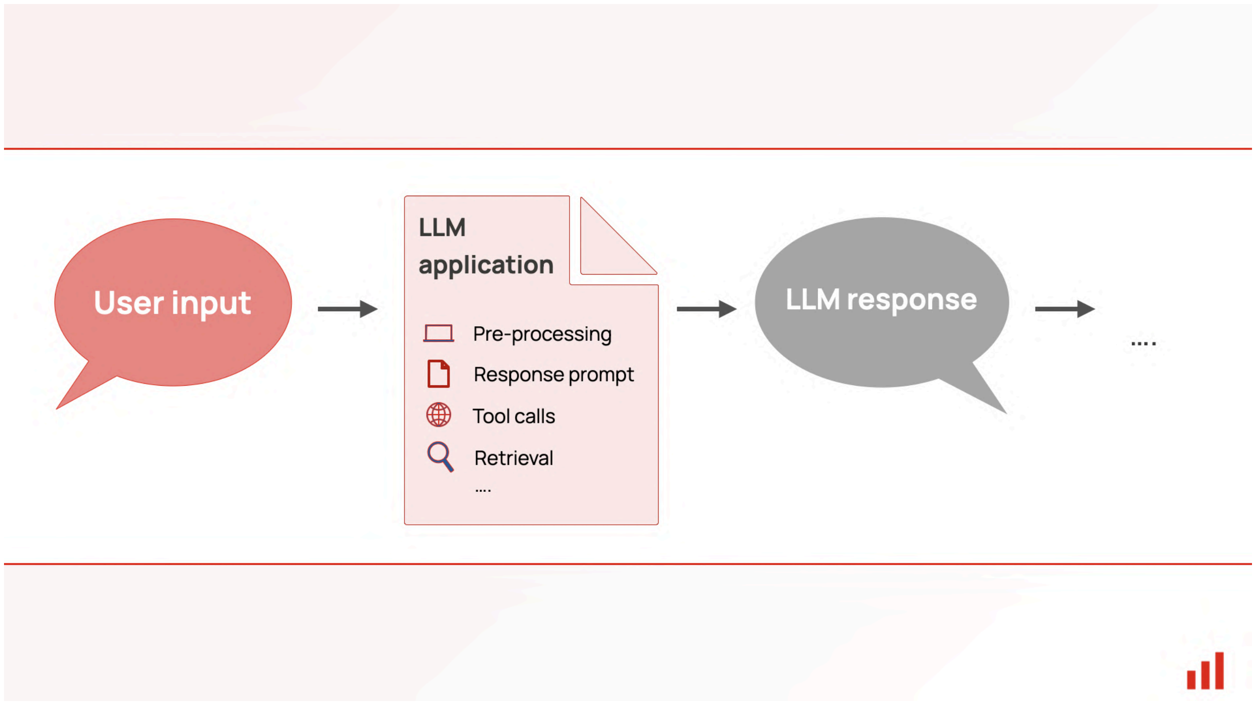
At first, it may seem odd to use an LLM to evaluate the text outputs. If the LLM is generating the answers, why would it be better at judging them or noticing errors?

The key is that we're not asking the LLM to review its work directly. Instead, we use a different prompt — or even a different LLM — to perform a separate task: evaluating specific properties of the text.

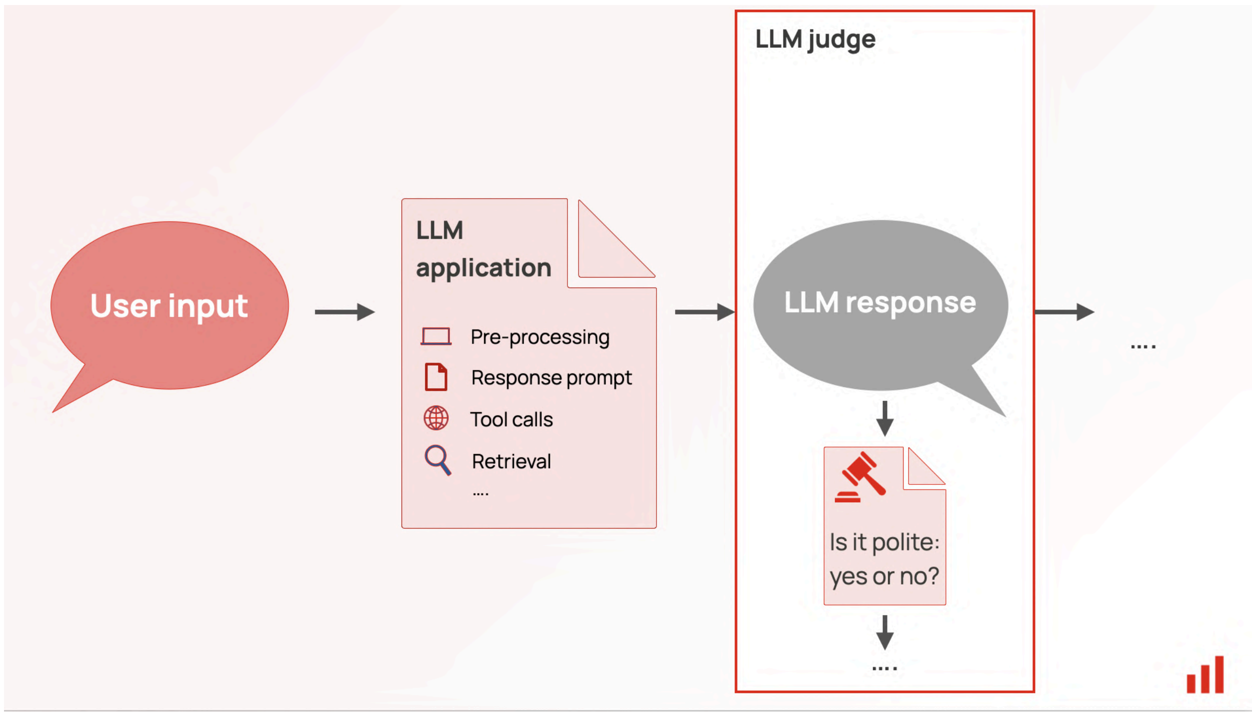
By making an external call to an LLM with a focused **evaluation prompt**, you activate different capabilities of the model. Often, using it as a simple text classifier and asking to follow a single instruction.

Think of it this way: it's easier to critique than to create. Classifying content is simpler than generating it. Detecting that something went wrong is usually easier than preventing the mistake in the first place.

When generating responses, an LLM handles many variables, integrates complex context and user inputs, and follows detailed product prompts that may contain multiple instructions at once. This complexity can lead to errors.



Evaluating responses is generally more straightforward. We're not asking the LLM to fix or correct itself; but simply to **assess** what has already been produced. For example, a **relevance evaluator** only needs to check whether the response is semantically relevant to the question, not generate a better answer.



Similarly, while your chatbot might be tricked by a malicious user into generating, say, biased content, an external LLM evaluator can still detect this. This evaluator works independently of the conversation: it examines the output and judges it on its merits. For example, "Does this text contain bias toward any group? Yes or no." Since LLMs are trained on vast amounts of text data, they are quite good at detecting language patterns.

This doesn't mean that LLM evaluators are "better" than your original model; they are simply being asked to do a simpler, more focused task.

Let's look at a few different types of judges you can implement.

Types of LLM judges

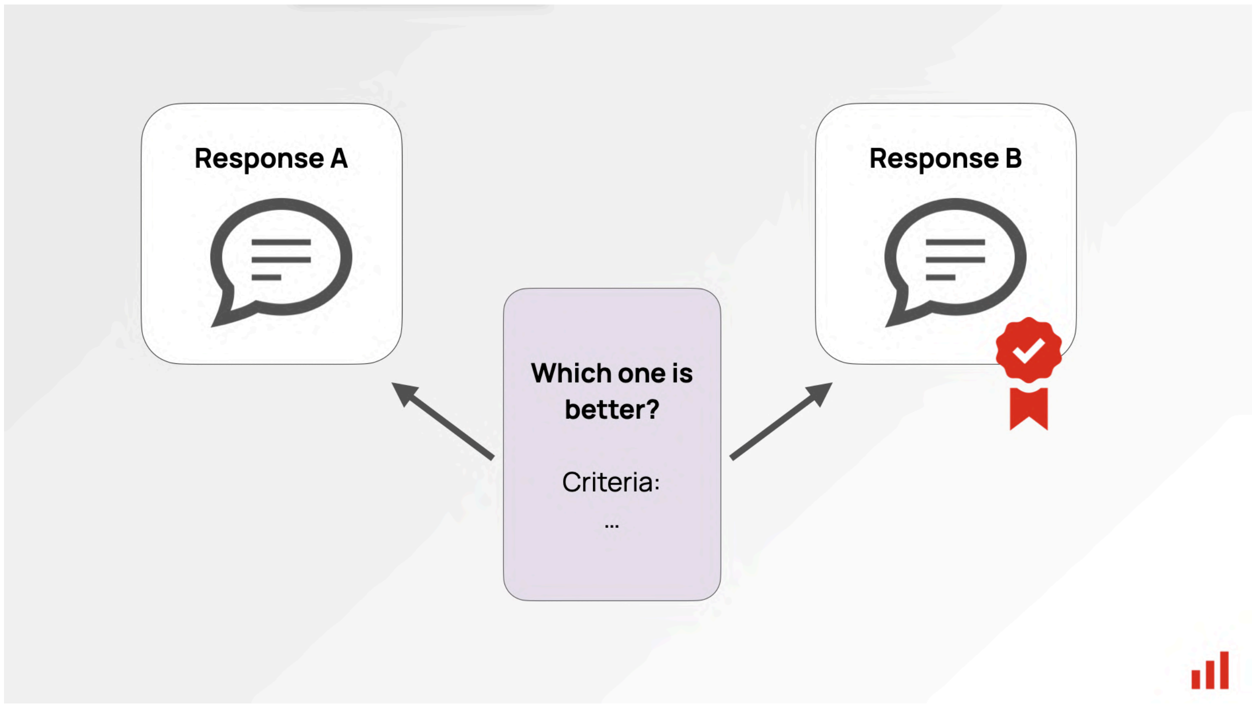
TL;DR: You can ask an LLM to choose the best answer between two, assess specific qualities like conciseness or politeness, or evaluate an answer with extra context, like to detect hallucinations or determine relevance.

You can use LLM evaluators in several scenarios:

- **Pairwise comparison**: give LLM two responses and ask to choose the better one. This lets you compare models, prompts, or configurations to see which performs best.
- **Evaluation by criteria (reference-free)**: ask the LLM to assess a response or conversation based on tone, clarity, correctness, or other dimensions.
- **Evaluation by criteria (reference-based)**: provide extra context, like a source document or reference, and ask the LLM to score the response.

Pairwise comparison is typically done offline: you need an option to generate and contrast multiple responses. Direct scoring works both offline and online for continuous monitoring.

Pairwise comparison



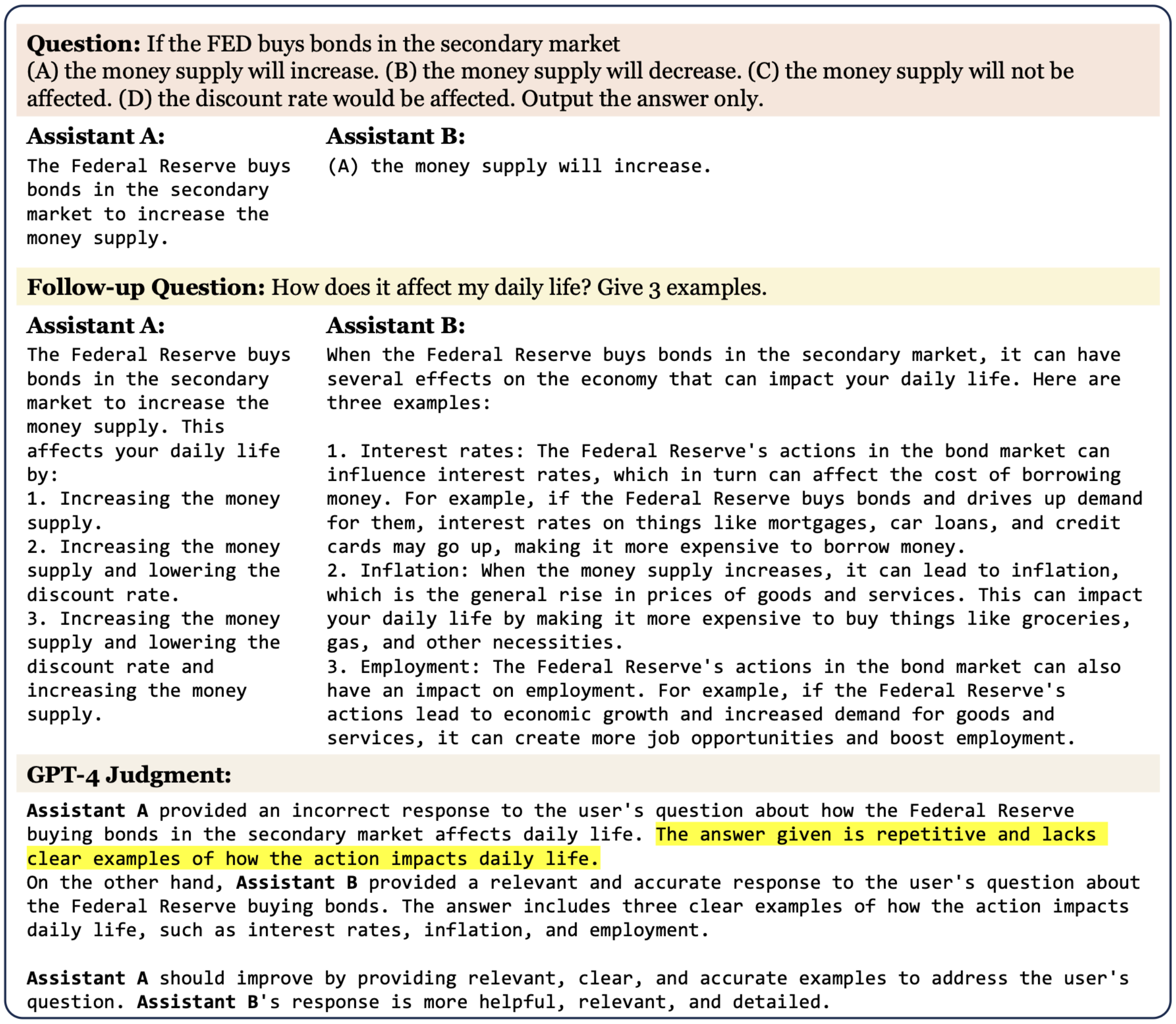
One of the early uses of the LLM-as-a-judge technique is in **pairwise comparison**, where the LLM looks at the two responses and decides which one is better. This method was described, for example, in this blog post, and got its official "judge" nickname in the paper "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena" by Zheng et al., 2023 (MT-Bench and Chatbot Arena are examples of LLM benchmarks).

In short, you generate two responses and ask the LLM to select the more appropriate one based on specific qualities or factors.

In the cited paper, the authors used GPT-4 as an evaluation model and compared its decisions to crowdsourced human preferences. The LLM evaluations achieved over 80% agreement, which is comparable to the level of agreement between human evaluators.

This is a great result, and the method has made its way into practitioners' playbooks. It is especially useful during the **development stage** when you want to pick the better model or prompt by running multiple pairwise comparisons.

Simplified prompt: *"You will be shown two responses to the same question. Your task is to decide which response is better based on the following criteria: relevance, helpfulness, and level of detail. If both responses are equally good, declare a tie."*



Question: If the FED buys bonds in the secondary market
(A) the money supply will increase. (B) the money supply will decrease. (C) the money supply will not be affected. (D) the discount rate would be affected. Output the answer only.

Assistant A:

```
The Federal Reserve buys
bonds in the secondary
market to increase the
money supply.
```

Assistant B:

```
(A) the money supply will increase.
```

Follow-up Question: How does it affect my daily life? Give 3 examples.

Assistant A:

```
The Federal Reserve buys
bonds in the secondary
market to increase the
money supply. This
affects your daily life
by:
1. Increasing the money
supply.
2. Increasing the money
supply and lowering the
discount rate.
3. Increasing the money
supply and lowering the
discount rate and
increasing the money
supply.
```

Assistant B:

```
When the Federal Reserve buys bonds in the secondary market, it can have
several effects on the economy that can impact your daily life. Here are
three examples:

1. Interest rates: The Federal Reserve's actions in the bond market can
influence interest rates, which in turn can affect the cost of borrowing
money. For example, if the Federal Reserve buys bonds and drives up demand
for them, interest rates on things like mortgages, car loans, and credit
cards may go up, making it more expensive to borrow money.
2. Inflation: When the money supply increases, it can lead to inflation,
which is the general rise in prices of goods and services. This can impact
your daily life by making it more expensive to buy things like groceries,
gas, and other necessities.
3. Employment: The Federal Reserve's actions in the bond market can also
have an impact on employment. For example, if the Federal Reserve's
actions lead to economic growth and increased demand for goods and
services, it can create more job opportunities and boost employment.
```

GPT-4 Judgment:

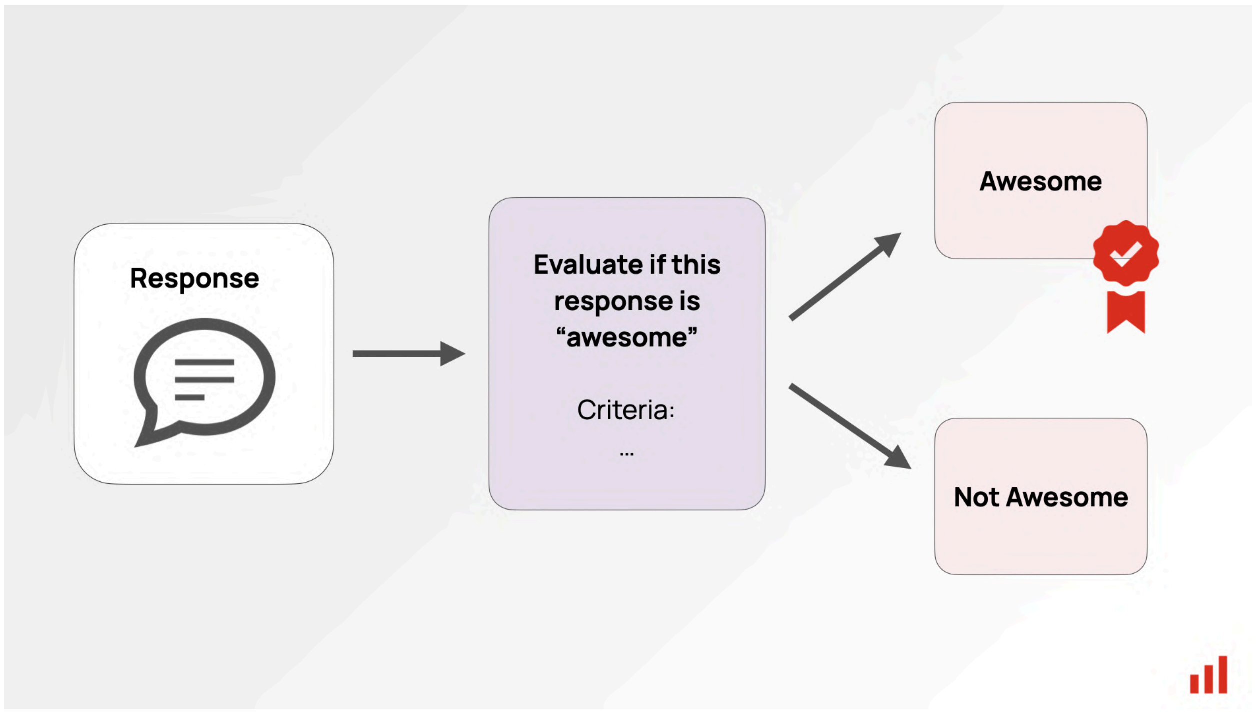
Assistant A provided an incorrect response to the user's question about how the Federal Reserve buying bonds in the secondary market affects daily life. The answer given is repetitive and lacks clear examples of how the action impacts daily life.
On the other hand, **Assistant B** provided a relevant and accurate response to the user's question about the Federal Reserve buying bonds. The answer includes three clear examples of how the action impacts daily life, such as interest rates, inflation, and employment.

Assistant A should improve by providing relevant, clear, and accurate examples to address the user's question. **Assistant B**'s response is more helpful, relevant, and detailed.

Example from "Judging LLM-as-a-Judge" paper.

The ability of LLMs to match human preferences when comparing two outputs is very promising. However, the more common use of LLM judges by LLM product builders is even simpler: direct scoring of individual responses.

Evaluation by criteria



You can ask the LLM to score the generated texts directly by any dimension you define. Instead of focusing on general "preference", it can handle one criterion at a time. This is particularly useful for production monitoring.

For example, you might use LLM judges to assess things like tone, clarity, adherence to format, conciseness, politeness, presence of Personally Identifiable Information (PII), etc.

These are generic examples. For your specific app, you'd likely develop custom evaluators that reflect what defines a quality output or what types of errors you expect to catch.

LLMs are highly effective at classification tasks that focus on language and semantics. A common way is to treat the evaluation as a **binary classification** problem. Alternatively, you can use a **grading scale** to measure how well the response meets specific criteria, such as using a Likert scale from one to five.

Simplified prompt: *"Evaluate the following text for conciseness. A concise response delivers the key message clearly and directly, without unnecessary words. Return one of the following labels: 'Concise,' or 'Verbose'."*

Research: The use of LLMs for grading is examined in papers like "Can Large Language Models Be an Alternative to Human Evaluation?" (Chiang et al., 2023), which compares LLM performance to human evaluators using the same instructions, "GPTScore: Evaluate as You Desire" (Fu et al., 2023), and "Is ChatGPT a Good NLG Evaluator? A Preliminary Study" (Wang et al.), which explores task-specific and aspect-specific grading evaluations.

You can also evaluate entire conversations. In this case, you'd pass the complete multi-turn transcript: all questions a user asked inside a given session and how LLM responded.

As long as this conversation transcript fits within the LLM **context window** (the amount of text it can process at once, typically a few thousand tokens), the model can handle this. It's still a classifier, just with a longer input text.

Examples of conversation-level evaluations are:

- **Detecting denials**: Does the agent refuse to complete a task at any point?
- **Identifying repetitions**: Does the user have to repeat their request?
- **Detecting user frustration**: Does the user express negative emotions?
- **Determining if the issue was resolved**: Was the user's problem solved by the end?

You can use it to monitor trends and flag individual conversations for human review.

Simplified prompt: *"Read the conversation and assess if the user's initial problem or request was resolved. 'Resolved' means the issue was addressed and the user confirmed or showed satisfaction. Return one of these labels: 'Resolved,' 'Not Resolved,' or 'Unknown'."*

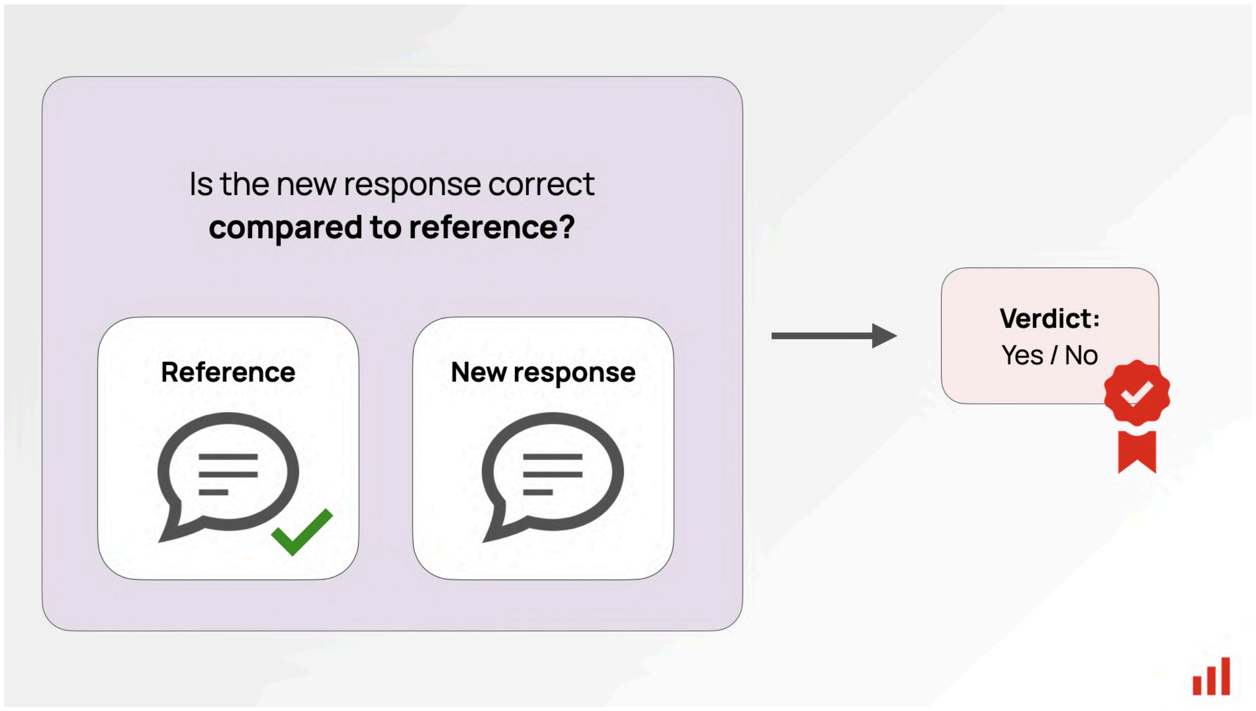
Reference-based evaluation

In **reference-based evaluations**, you don't just evaluate the generated response alone—you provide additional context to review. Here are a few examples of extra inputs:

- **Answer + Reference Answer**: Useful when you have a ground truth or correct answer to compare against.
- **Answer + Question**: Common in chatbot or Q&A systems to check if the response properly addresses the question.
- **Answer + Retrieved Context** or **Question + Retrieved Context**: Used in Retrieval-Augmented Generation (RAG), when you first look for documents to back up the answer.

You still directly score the response, but instead of passing it as a single input, you include two or more pieces of text in the evaluation prompt and explain how they relate to each other.

1. Evaluating correctness based on reference answer



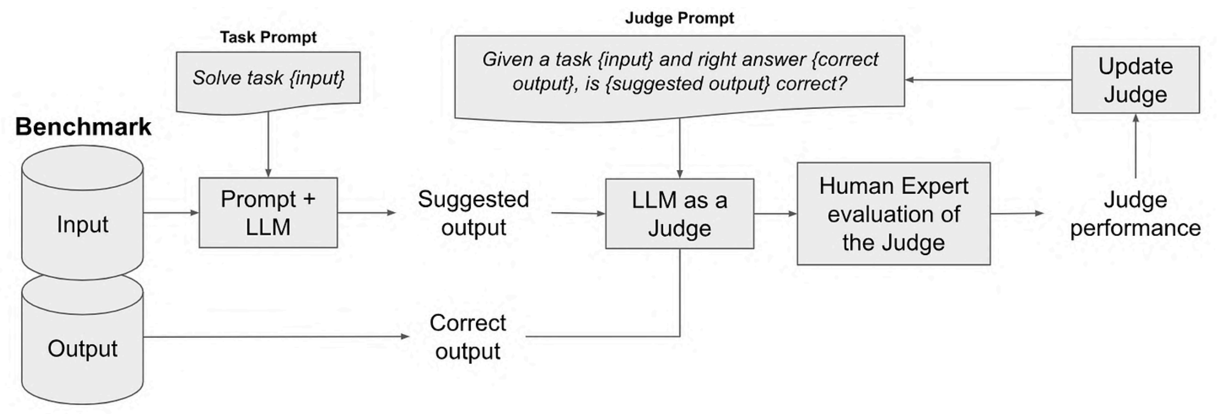
This is an offline evaluation where the LLM compares a generated response to a "golden" reference answer. It's great for evaluations during the experimental phase (when you iterate on your system design) and for regression testing after updates to the model or prompt.

For example, in a Q&A system, the LLM judge can check if the new response is similar to the approved reference answer.

This **correctness judge** is an alternative to deterministic metrics like **ROUGE** that quantify word and phrase overlap between two answers, and **semantic similarity** checks, which perform the comparison using a pre-trained embedding model.

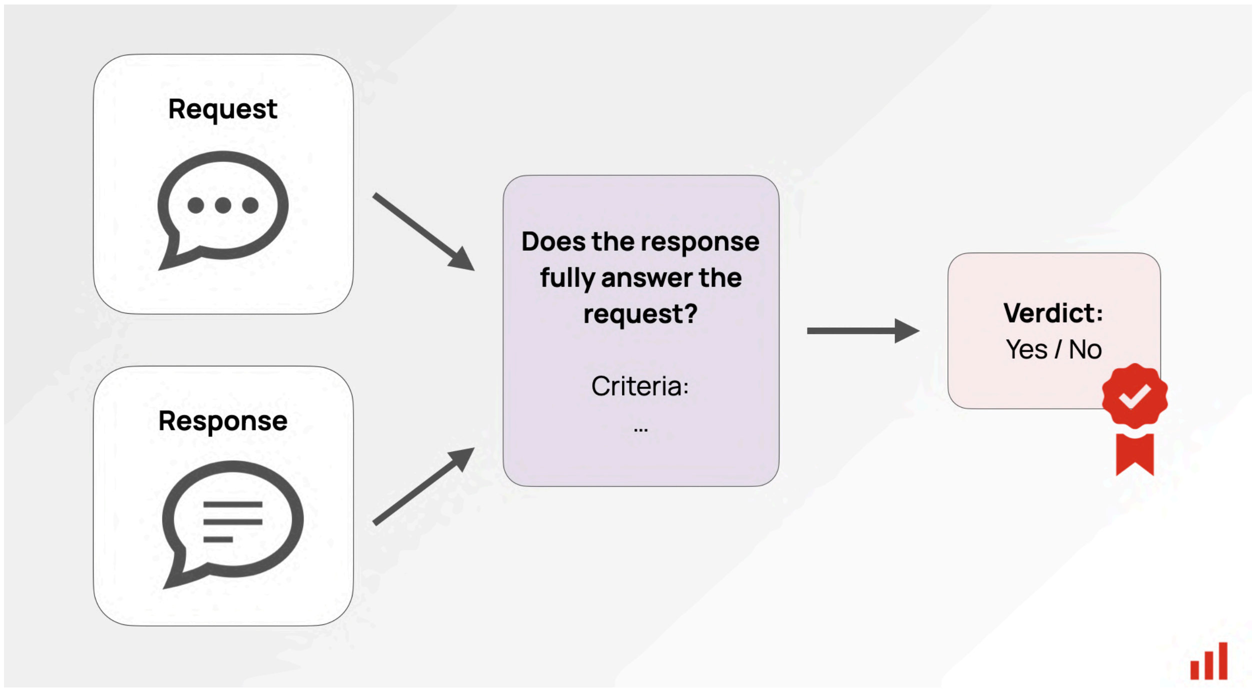
Simplified prompt: *"Compare the generated RESPONSE to the REFERENCE answer. Evaluate if the generated response correctly conveys the same meaning as the reference, even if the wording is different. Return one of these labels: 'Correct' or 'Incorrect.'"*

Examples: In Segment's blog post, they used an LLM evaluator to compare the LLM-generated queries against a pre-approved query for a given input. Similarly, Wix describes how they use LLM judges to match the correctness of responses in a Q&A use case against ground truth.



An example from the Wix blog.

2. Evaluating answer quality considering the question



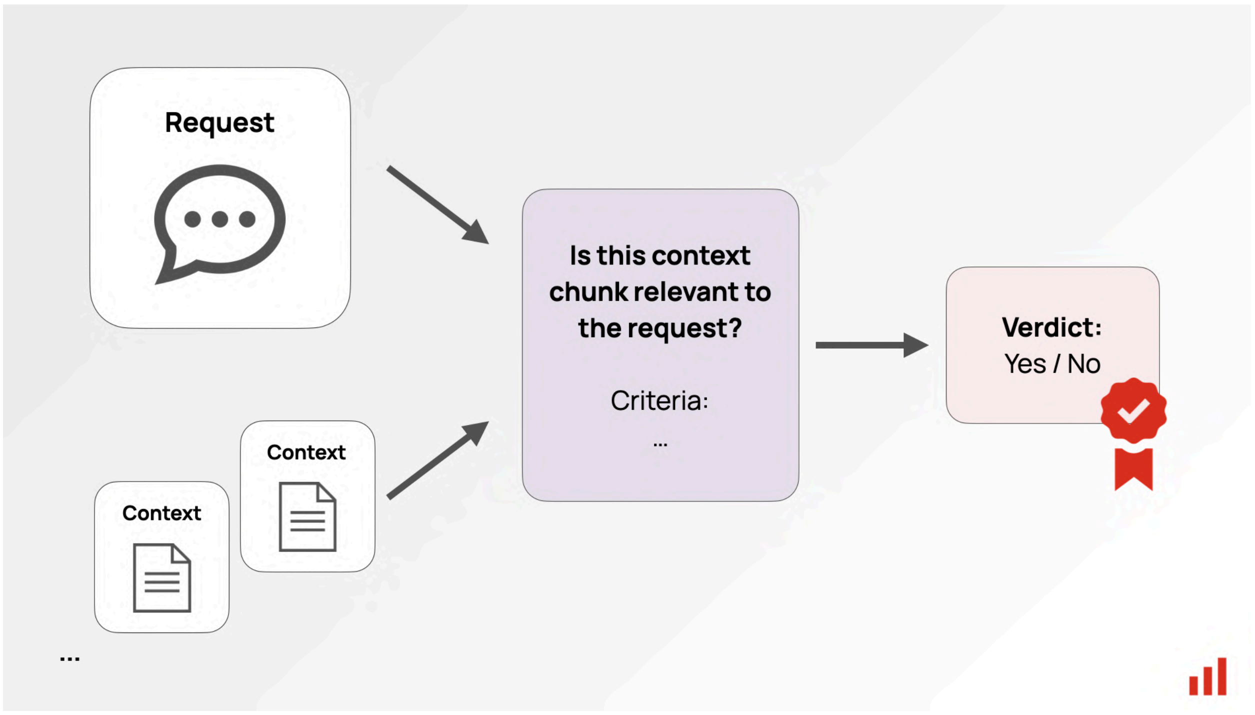
Unlike the previous example that requires a golden reference answer, you can also run online evaluations using additional data available at the time of response generation. For instance, in Q&A systems you can judge the answer together with the question to evaluate:

- **Completeness**: Does the answer fully address the question?
- **Relevance**: Is the answer directly related to the question asked?

You can run these types of evaluations for ongoing quality monitoring.

Simplified prompt: *"Evaluate the following RESPONSE based on the given QUESTION. A complete response is one that fully addresses all parts of the question. Return one of the following labels: 'Complete' or 'Incomplete."*

3. **Scoring context relevance in RAG.**



Retrieval-Augmented Generation (RAG) is a special case of LLM product architecture.

With this implementation, the system first searches for documents that can help answer the question and then uses them to generate a response. This helps add up-to-date knowledge to the LLM response. To properly evaluate RAG's performance, you need to assess both sides:

- How well it retrieves relevant documents.
- How good the final answer is.

For the first part — evaluation of search quality — you can use ranking quality metrics like NDCG or precision at K. These types of metrics quantify how well the system can find and sort the documents that help answer the query. These evaluations typically happen offline as you iterate on parameters like different search strategies.

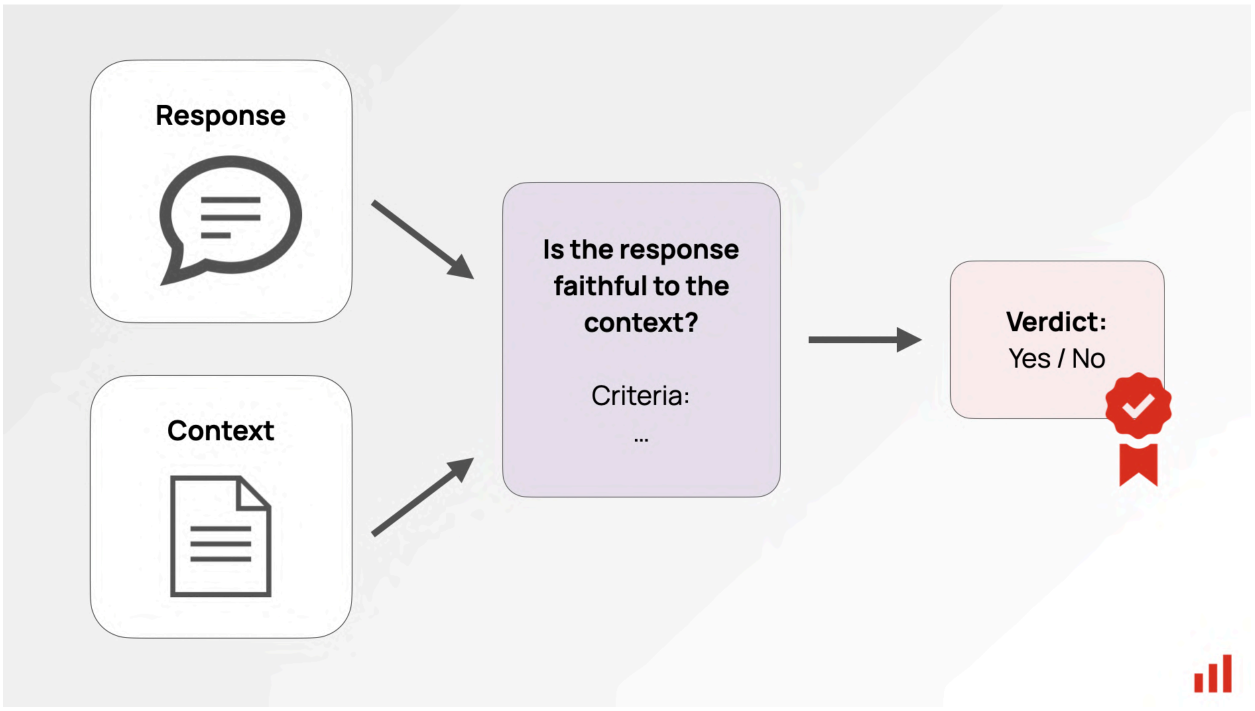
However, ranking metrics require **relevance labels**, meaning each document must be marked as helpful or not for answering the question. You can do this manually, but another option is to task an LLM. In this case, it will act as a **context relevance** judge.

After the LLM scores each retrieved text chunk for its relevance to the query, you feed these labels onto the next step of the evaluation process to compute the ranking quality metrics.

Simplified prompt: *"Evaluate the relevance of the following TEXT in answering the given QUESTION. A relevant text contains information that helps to answer the question, even if partially. Return one of the following labels: 'Relevant', or 'Irrelevant.' "*

For example, this method is described in the paper "Large Language Models Can Accurately Predict Searcher Preferences" (Thomas et al., 2023).

4. Evaluating hallucinations in RAG



On the other side of the equation, and especially once the RAG system is in production, you want to check the final response quality. One issue that happens here is hallucination: LLMs can invent details not present in the source material.

Since you have both the **answer** and the **retrieved context,** you can run one more evaluation to see how grounded the answer is. Effectively, you create a **faithfulness judge** to double-check if the LLM processed the retrieved content correctly.

Simplified prompt: *"Evaluate the following RESPONSE for faithfulness to the CONTEXT. A faithful response should only include information present in the CONTEXT and not invent new details. Return one of the following labels: 'Faithful' or 'Not Faithful.'"*

You can also use this approach to evaluate summaries by comparing them to the source. This helps you cross-check the quality and spot inconsistencies. For example, this paper explores methods to detect factual inaccuracies in summaries.

Example: In DoorDash's blog post about their RAG-based support agent, they mention using LLM judges to evaluate response coherence with context.

LLM judge best practices

LLM evaluators are a powerful technique, but they require careful setup for your specific scenario. You'll need time to design and evaluate your judge, craft good prompts, and build a reliable evaluation pipeline for continuous monitoring.

Let's explore some of these best practices.

How to create an LLM judge

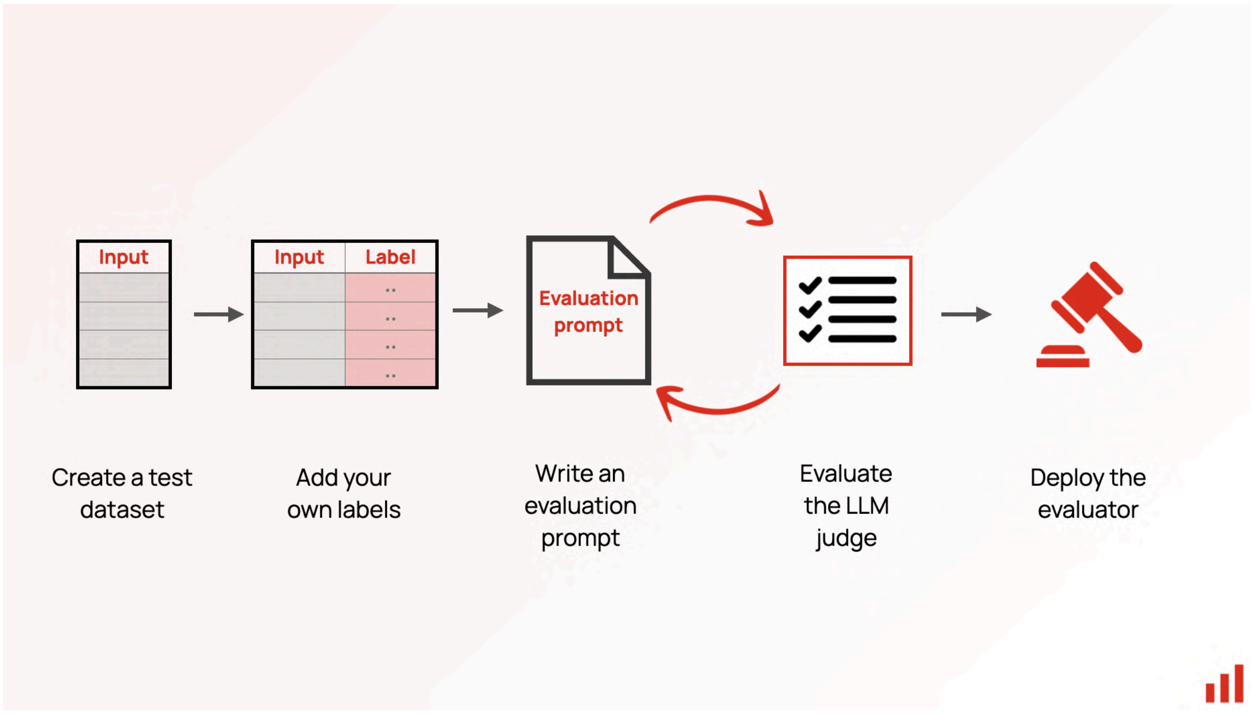
TL;DR: Creating an LLM judge is a small ML project. Start with a labeled dataset that reflects how you want the LLM to judge your texts, then create and refine the evaluation prompt to ensure alignment with your labels.

Creating an LLM judge is much like developing any LLM-powered product — you need a prompt that tells the LLM exactly what to do. In this case, it's an **evaluation prompt** that instructs the LLM to assess text inputs and return a label, score, or explanation.

Here's the rub: if you're using an LLM to evaluate other LLMs, and the results aren't deterministic, how do you ensure your judges align with your expectations?

You'll need to take an iterative approach — refining the judge just like you refine your LLM product prompts. In other words, your evaluation system needs its own evaluation!

Here is the general process.



Step 1. Define the evaluation scenario.

First, decide what exactly you want the LLM judge to evaluate. Are you checking for correctness, tone, conciseness, or something else? You might evaluate just the generated output, compare multiple inputs, or look at the complete conversation transcript. Start with what you want to achieve, then think back to how to define the evaluator.

Tip: Keep it simple! Don't try to evaluate too many things at once. If you want to check different dimensions (like tone and accuracy), split them into separate evaluations. Use clear, binary choices whenever possible (e.g., "correct" vs. "incorrect").

Step 2. Prepare the evaluation dataset.

Next, create a small dataset to test your LLM judge. This can include examples from experiments or production data. If you don't have those, you can create synthetic cases that mimic the expected inputs.

Your dataset doesn't need to be huge, but it should include diverse examples, especially those that challenge your evaluation criteria.

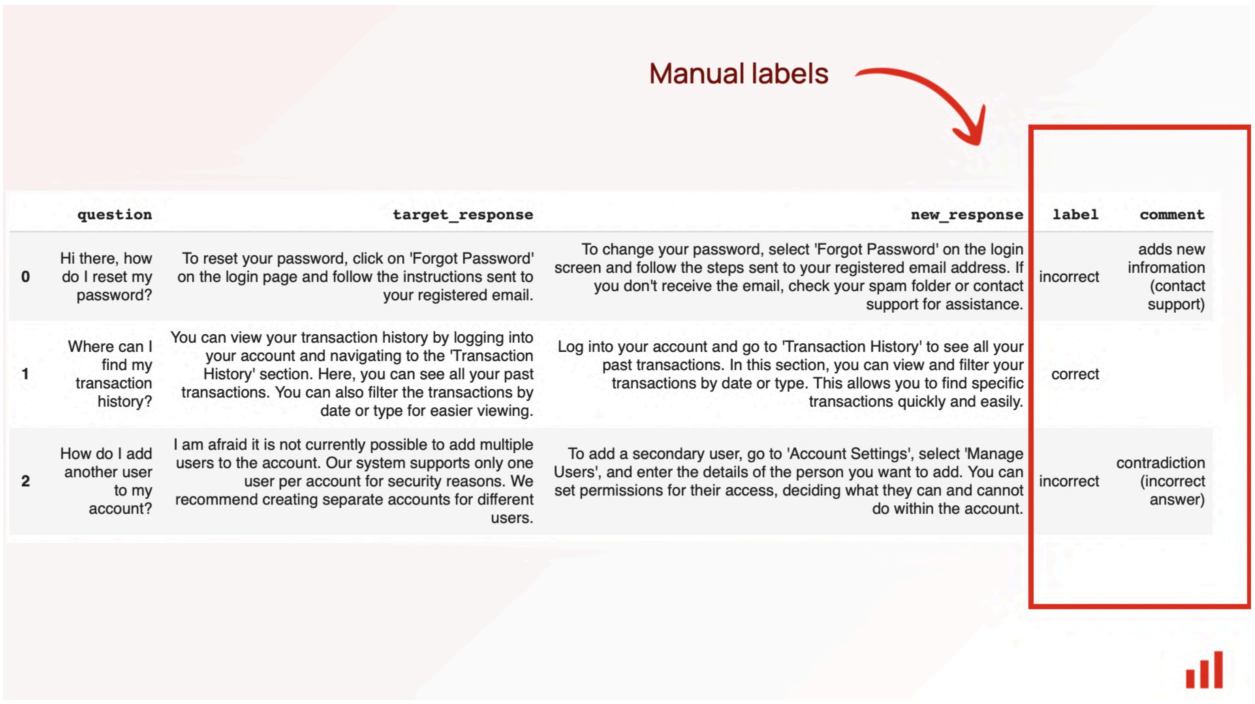
Step 3. Label this dataset.

Here's the important part: you need to **label this dataset manually**, just as you want the LLM judge to do later. This labeled dataset will be your "ground truth" and help you measure how well the LLM judge performs.

Plus, labeling it yourself forces you to be intentional about what you expect the LLM to notice. This will shape how you write the evaluation instructions.

Let's say you are creating a correctness judge that will compare two LLM outputs — such as an old vs. new answer after a model update. You'll need to decide which changes matter: are you checking if the responses are mostly the same, or do specific details count? Maybe some changes, like adding a greeting, are fine, but suggesting an extra action might be unacceptable. Often, you'd look for specific types of errors like contradictions, omissions, changes of order, or response style.

For example, in this use case we considered any extra information, such as suggesting the user contact an agent, as incorrect, even if it didn't contradict the reference answer directly.



	question	target_response	new_response	label	comment
0	Hi there, how do I reset my password?	To reset your password, click on 'Forgot Password' on the login page and follow the instructions sent to your registered email.	To change your password, select 'Forgot Password' on the login screen and follow the steps sent to your registered email address. If you don't receive the email, check your spam folder or contact support for assistance.	incorrect	adds new infromation (contact support)
1	Where can I find my transaction history?	You can view your transaction history by logging into your account and navigating to the 'Transaction History' section. Here, you can see all your past transactions. You can also filter the transactions by date or type for easier viewing.	Log into your account and go to 'Transaction History' to see all your past transactions. In this section, you can view and filter your transactions by date or type. This allows you to find specific transactions quickly and easily.	correct	
2	How do I add another user to my account?	I am afraid it is not currently possible to add multiple users to the account. Our system supports only one user per account for security reasons. We recommend creating separate accounts for different users.	To add a secondary user, go to 'Account Settings', select 'Manage Users', and enter the details of the person you want to add. You can set permissions for their access, deciding what they can and cannot do within the account.	incorrect	contradiction (incorrect answer)

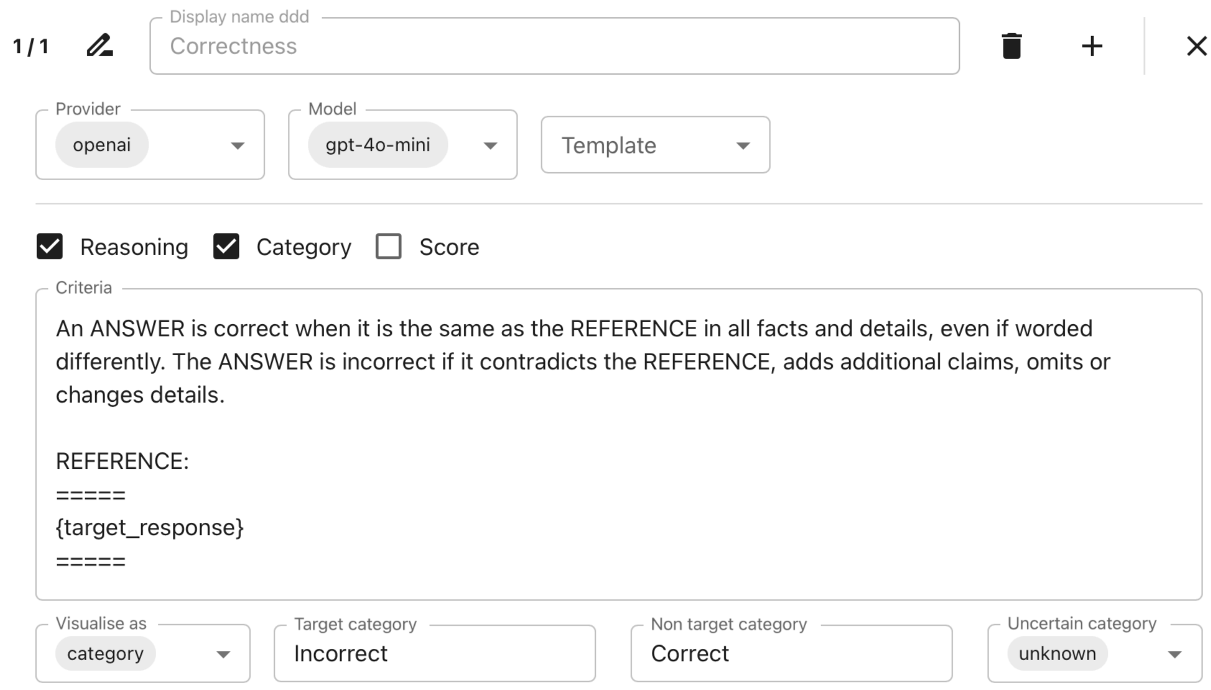
This may be different for your use case!

You can, of course, skip this step: write the best prompt you can, then review and correct the LLM-generated labels to create the approved test dataset. However, starting with your labels often improves your initial prompt.

Step 4. Craft your evaluation prompt.

Once you know what you're looking for, it's time to create an evaluation prompt for the LLM.

For example, here's how we formulated the evaluation criteria after manually labeling the data for the correctness example. It accounts for specific errors we've seen:


1 / 1

Display name ddd: Correctness

Provider: openai

Model: gpt-4o-mini

Template

- [x] Reasoning
- [x] Category
- [ ] Score

Criteria:

An ANSWER is correct when it is the same as the REFERENCE in all facts and details, even if worded differently. The ANSWER is incorrect if it contradicts the REFERENCE, adds additional claims, omits or changes details.

REFERENCE:
=====
{target_response}
=====

Visualise as: category

Target category: Incorrect

Non target category: Correct

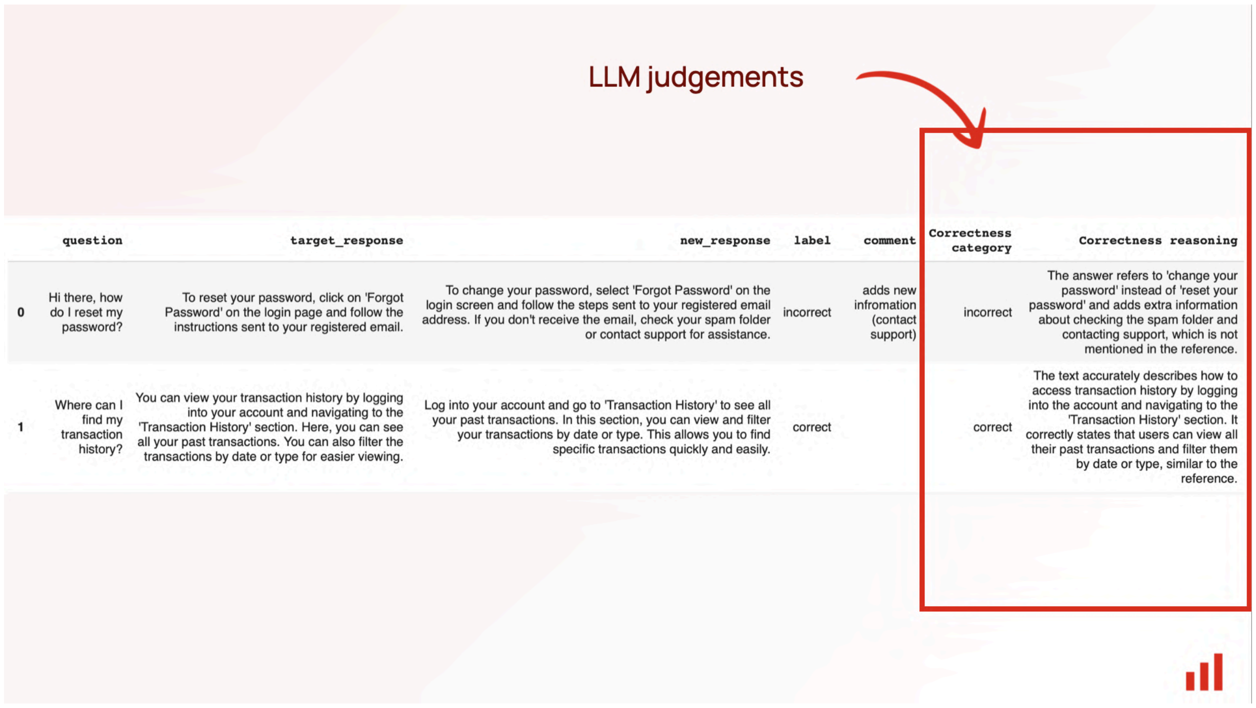
Uncertain category: unknown

We'll share more tips on prompt writing in the next section.

Want a code tutorial? Check out this guide for a step-by-step example of creating an evaluation dataset and designing an LLM judge.

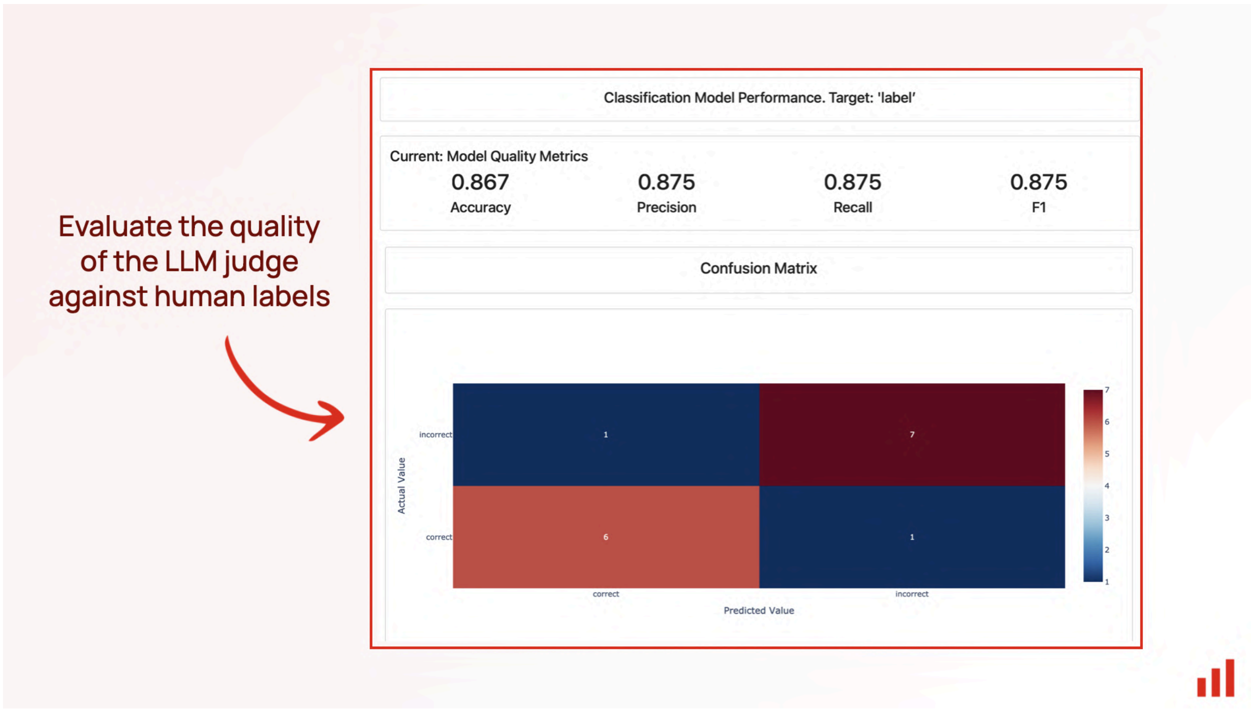
Step 5. Evaluate and iterate.

Once your prompt is ready, apply it to your evaluation dataset and compare the LLM judge's outputs to your manually labeled ground truth.



	question	target_response	new_response	label	comment	Correctness category	Correctness reasoning
0	Hi there, how do I reset my password?	To reset your password, click on 'Forgot Password' on the login page and follow the instructions sent to your registered email.	To change your password, select 'Forgot Password' on the login screen and follow the steps sent to your registered email address. If you don't receive the email, check your spam folder or contact support for assistance.	incorrect	adds new infromation (contact support)	incorrect	The answer refers to 'change your password' instead of 'reset your password' and adds extra information about checking the spam folder and contacting support, which is not mentioned in the reference.
1	Where can I find my transaction history?	You can view your transaction history by logging into your account and navigating to the 'Transaction History' section. Here, you can see all your past transactions. You can also filter the transactions by date or type for easier viewing.	Log into your account and go to 'Transaction History' to see all your past transactions. In this section, you can view and filter your transactions by date or type. This allows you to find specific transactions quickly and easily.	correct		correct	The text accurately describes how to access transaction history by logging into the account and navigating to the 'Transaction History' section. It correctly states that users can view all their past transactions and filter them by date or type, similar to the reference.

For binary classifications, you can use metrics like [precision and recall] to measure how well the LLM judge is performing, especially when you're focusing on a specific class. For example, you might prioritize the LLM's ability to detect "incorrect" responses, even if some correct responses are misclassified: in this case, [recall] is a relevant quality metric.



Classification Model Performance. Target: 'label'

Current: Model Quality Metrics

Accuracy	Precision	Recall	F1
0.867	0.875	0.875	0.875

If the results don't meet your expectations, you can adjust the prompt and rerun the evaluation. Ideally, you would also keep some part of your manually labeled dataset as a "held-out" dataset: and only use it to test your final prompt.

Note that your LLM judge doesn't need to be perfect — just "good enough" for your goals. Even human evaluators make mistakes!

And finally, bring in the domain experts.

Non-technical team members — like product or subject matter experts — play a big role in setting the evaluations. They help figure out what behaviors or topics the LLM should catch, shape guidelines that go into the prompts, and test the alignment. With no-code tools, you can design this a collaborative workflow.

Creating LLM judges is also naturally an iterative process. Especially if you begin with a synthetic dataset, you might need to adjust your standards once you start grading real user data and see patterns in the LLM outputs and errors. You may then edit the judge prompt or split and add new evaluation criteria.

Research: In the paper “Who Validates the Validators?” (Shankar et al., 2024), the authors talk about "criteria drift," where manually grading outputs helps users refine their expectations based on the specific LLM outputs they see.

LLM evaluation prompts

TL;DR: Write your own prompts. Use yes/no questions and break down complex criteria. Asking for reasoning helps improve evaluation quality and debugging.

Evaluation prompts are the core of your LLM judge. Just like with human assessors, you need to provide the LLM with precise, detailed instructions.

Ideally, you should write your own evaluation prompts. First, because this is where LLM judges really shine — you can customize them to your specific needs. Second, even minor clarifications can improve the quality of your scenario over a generic prompt.

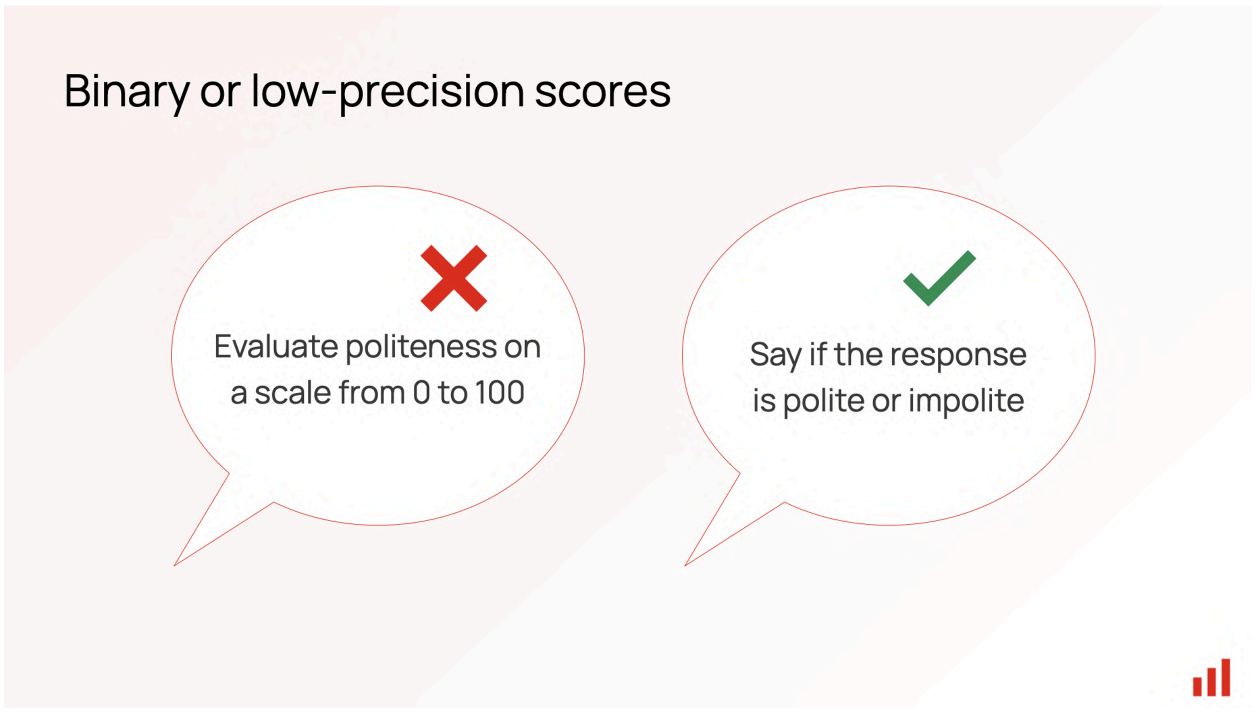
Keep in mind that prompts behave differently across models. The exact evaluation instructions might give different results on a GPT-4.0 mini compared to GPT-4.0. Testing them against your expectations is key.

Tip: If you’re using external evaluation tools with LLM-based metrics, always review their prompts and test against your labels to ensure they align with your needs.

Several prompting techniques can improve the accuracy and consistency of your LLM evaluator. These are similar to the ones you might use while developing your LLM product, such as a chain of thought prompting. Asking LLMs to grade text is no different from other tasks you might ask them to perform — the same tricks work.

Here are a few ones to consider.

Use binary or low-precision scoring.


Binary or low-precision scores
Evaluate politeness on a scale from 0 to 100
Say if the response is polite or impolite

Binary evaluations, like "Polite" vs. "Impolite," tend to be more reliable and consistent for both LLMs and human evaluators. It's easier to get accurate results with two simple choices rather than trying to decide if a specific response scores 73 vs. 82 for "politeness."

While purpose-built probabilistic machine learning models can return such scores, LLMs generate text and aren't naturally calibrated for high-precision scoring.

You can also use a three-option approach, like "relevant," "irrelevant," and "partially relevant," or include an "unknown" option when there's not enough information. This avoids forcing the LLM to make a decision without sufficient data.

Explain the meaning of each score.

Think of writing prompts like giving instructions to an intern who is doing the task for the first time. Don't just ask the LLM to label something as "toxic" or "not toxic". Instead, clearly define what "toxic" means — like language that's harmful or offensive in specific ways.

If you prefer to flag borderline cases rather than miss them, you can guide the LLM to be more strict with instructions like, "If unsure, err on the side of caution and mark it as 'toxic'".

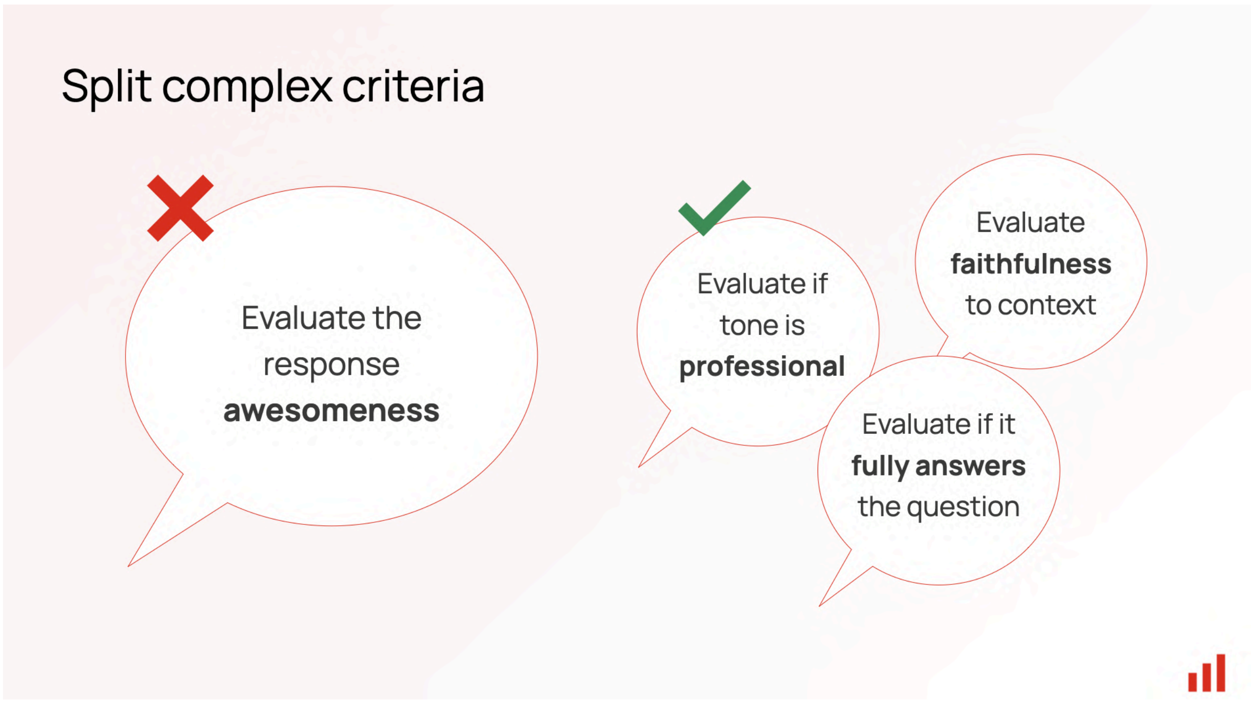
Explaining what each class means becomes even more critical with something like a 5-point scale — what's the difference between a 3 and a 4? If that's not clear, both LLMs and human reviewers will struggle to stay consistent.

If you can't easily explain these distinctions, it might be a sign you need to simplify the scale. Without guidance, LLMs may return inconsistent results, giving different scores for similar texts or leaning toward certain scores that were more common in its training data.

Using multiple judgments: To reduce variability, you can use multiple models or repeat evaluations, then combine the results using methods like max voting or averaging. For example, see "*Replacing Judges with Juries*" (Verga et al., 2024).

Want to check if your instructions are good? Try labeling some responses yourself! If you need extra clarification to make a decision, the LLM will likely do so too. If you're adding too many details, consider breaking the task into several.

Simplify evaluation by splitting criteria.


Split complex criteria
Evaluate the response **awesomeness**
Evaluate if tone is **professional**
Evaluate **faithfulness** to context
Evaluate if it **fully answers** the question

If you have several aspects to evaluate, like completeness, accuracy, and relevance, it's best to split them into separate evaluators. This keeps things focused.

You can still combine the results for an overall judgment in a deterministic way. For instance, you could flag an answer if any one criterion gets a "negative" label (incomplete, inaccurate, or irrelevant). Alternatively, you can sum up the number of "good" labels for an overall score or assign weights to reflect the importance of each criterion.

This way, LLM functions as a simple text classifier, handling one quality at a time rather than dealing with complex reasoning all at once. This improves the accuracy of each evaluation and also makes it easier for reviewers to verify and understand the results.

Add examples to the prompt.

If your criteria is nuanced, you can also consider adding examples of inputs and judgments.

Your prompt can start with a general instruction (e.g., "Good" means... "Bad" means...) and then provide examples of both good and bad responses. These examples help the LLM better understand how to apply your criteria — especially helpful for less capable models.

Research: This technique is known as *few-shot learning*. For more details, refer to Brown et al., 2020, in their paper *Language Models are Few-Shot Learners*.

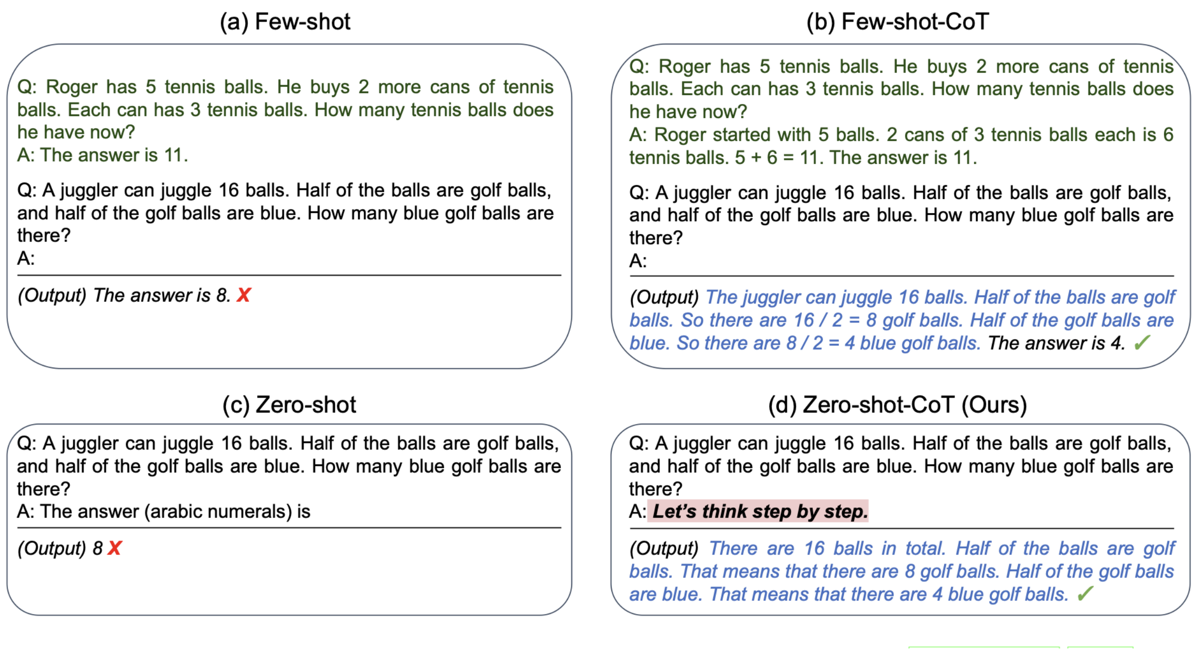
However, it's important to test how adding examples impacts the judge's performance. Skewed or biased examples can influence the LLM's decisions. For instance, if you include more negative examples than positive ones, or if all the negative examples are listed towards the end, their order or frequency may affect evaluation results. Though this tends to be less of an issue with more advanced models.

Research: For more on this, check the paper *Calibrate Before Use: Improving Few-Shot Performance of Language Models* (Zhao et al., 2021), where they show that few-shot learning results vary due to factors like prompt format and example order.

Encourage step-by-step reasoning.

Just like with other tasks, asking the LLM to "think" through its process before giving a final answer — known as the **Chain of Thought** (CoT) approach — can help achieve better results.

Research: The idea of adding a reasoning step inside a prompt is explored by Wei et al. (2022) in their paper *Chain-of-Thought Prompting*, and Kojima et al. (2022) in their paper *Large Language Models are Zero-Shot Reasoners*.



(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Examples of different Chain-of-Thought prompts from the paper by Kojima et al.

You can do the same in your evaluation prompt: ask the model to explain its reasoning or think step by step, effectively implementing a **Zero-Shot-CoT** approach. This way, the model will provide both the reasoning and the result in one response. Further research shows that this significantly improves the quality of evaluations.

In addition, this also creates a reasoning trail that you can review later. This is especially helpful for troubleshooting when you go through the responses. For example, the generated justification can highlight which parts of the text led the model to flag something as incorrect or identify that it contains personally identifiable information.

Multi-turn chain of thought. Researchers explore more sophisticated approaches to CoT. For example, one method, called **G-Eval** (see paper by Liu et al., 2023), uses a process where the AI first defines the task, plans the steps, and then completes the evaluation form. However, further studies show that this auto-generated CoT doesn't always work better, and simply asking LLM to explain or analyze outperforms this method (see Chiang et al., 2023).

Set a low temperature.

In LLMs, temperature controls how random the output is. Higher temperatures mean more variety, while lower temperatures make outputs more "predictable". For evaluations, you don't need creativity — set a low temperature so the model gives consistent answers for the same input.

Use a more capable model.

When evaluating, it makes sense to start with a stronger model. This generally helps ensure better alignment with human judgments. Once you have that solid baseline, you can experiment with smaller or less capable models to see if they meet your needs.

Get structured outputs.

Last but not least, always go for a structured output format, like JSON. It makes it much easier to parse the evaluation results for further analysis.

Recap

To sum up, here are a few tips for writing better evaluation prompts:

- Use binary or low-precision scores.
- Split complex criteria into separate evaluators.
- Clarify the meaning of each score or label, and consider adding examples.
- Ask the LLM to think step by step and provide reasoning along with the verdict.
- Set a low temperature.
- Use a more capable model when possible.

Starting with simpler approaches is often best. As long as you evaluate your LLM judges to confirm how well they work, you can begin with a straightforward method and stick with it if it's effective. If you want to experiment with more complex techniques (like a multi-step chain of thought with extra LLM calls), you can build upon this initial baseline.

For a more in-depth literature review on LLM-as-a-judge research, check out these excellent resources by Cameron Wolfe and Eugene Yan.

Want to get started with example prompts? We implemented many of these tips in the Evidently open-source library. It offers prompt templates for LLM judges that already include repeatable parts like formatting and reasoning. You can add your evaluation criteria in plain text or use built-in prompts for inspiration. Check out the docs to get started. To create judges with no-code, sign up for a free account on Evidently Cloud.

LLM observability in production

TL;DR: LLM observability helps monitor the performance of your system in production. Set up tracing to collect live data, schedule evaluations where portions of this data are scored by LLM judges, and use a dashboard to track trends such as user frustration and hallucinations.

Once your system goes live, real users will interact with it in ways you might not have expected. Even the most thorough pre-launch testing won't cover all the different ways people will use it. That's why it's so important to track actual performance in real time.

In production, there's no perfect answer to which to compare outputs, so you'll need to monitor the quality of responses on their own. LLM evaluators make this possible. You can set up a regular process to score new generations based on selected criteria.

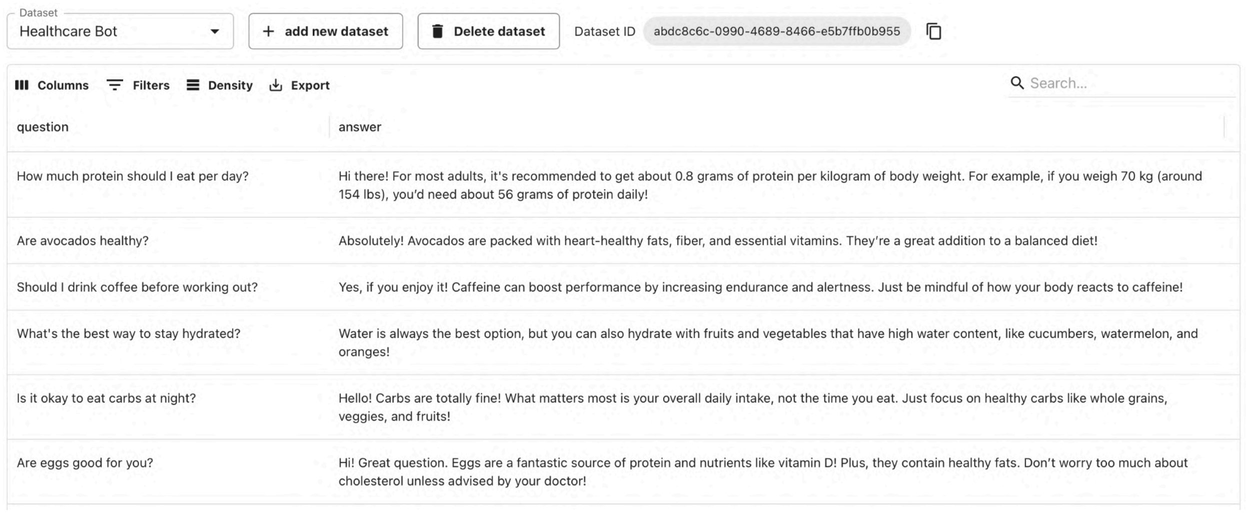
This monitoring isn't just about quality control — it can also provide insights into how users interact with your tool, such as what tasks or topics are most common.

Here's how you can set up the process:

1. Tracing.

The first step is **tracing** — collecting data from user interactions and storing it for analysis. You'll need to instrument your AI application to capture all inputs, LLM and function calls and completions.

The immediate benefit is clear: once you have the logs, you can view and read them to understand what's happening as users interact with your application.



question	answer
How much protein should I eat per day?	Hi there! For most adults, it's recommended to get about 0.8 grams of protein per kilogram of body weight. For example, if you weigh 70 kg (around 154 lbs), you'd need about 56 grams of protein daily!
Are avocados healthy?	Absolutely! Avocados are packed with heart-healthy fats, fiber, and essential vitamins. They're a great addition to a balanced diet!
Should I drink coffee before working out?	Yes, if you enjoy it! Caffeine can boost performance by increasing endurance and alertness. Just be mindful of how your body reacts to caffeine!
What's the best way to stay hydrated?	Water is always the best option, but you can also hydrate with fruits and vegetables that have high water content, like cucumbers, watermelon, and oranges!
Is it okay to eat carbs at night?	Hello! Carbs are totally fine! What matters most is your overall daily intake, not the time you eat. Just focus on healthy carbs like whole grains, veggies, and fruits!
Are eggs good for you?	Hi! Great question. Eggs are a fantastic source of protein and nutrients like vitamin D! Plus, they contain healthy fats. Don't worry too much about cholesterol unless advised by your doctor!

Initially, you might manually review responses to spot common patterns or issues. However, as the volume of data grows, manual reviews won't scale, so you'll need some automation.

2. Schedule evaluations.

Once you've got your tracing data stored and know what you want to evaluate, you can set up **scheduled evaluations**. These are regular checks where you run new answers or complete conversations through the LLM judge to see how the system's doing based on the attributes you've set. If you're handling a lot of data, you can sample a subset to track performance.

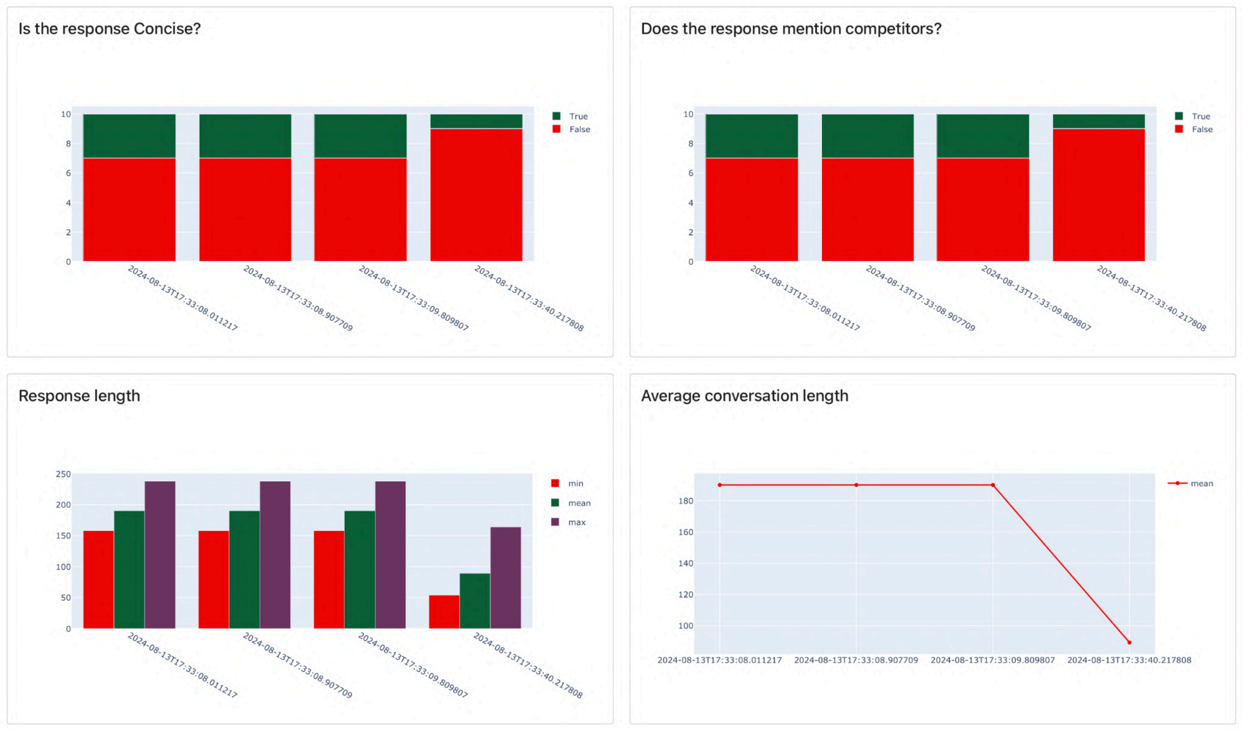
For example, if you're running a customer service chatbot, you could evaluate 10% of the conversations for signs of frustration expressed by the user, repeated questions, or unresolved chats. Running these evaluations regularly (e.g., every hour, day, or after X conversations) will keep you updated on performance.

You can also combine LLM evaluators with other methods, like using regular expressions to check for specific terms, such as product or competitor mentions.

3. Build a dashboard.

To make performance tracking easier, you will need a **dashboard.**

After running an evaluation on the latest batch of data, you can add metrics like the “share of answers labeled as hallucinated” or the “number of conversations where users expressed frustration” to a dashboard and visualize them over time. This helps track performance trends and spot any issues.



You can also set up **alerts** so that if things go off track, you’re notified right away and can step in before the issue affects too many users.

4. Look at your data!

Monitoring and debugging go hand in hand. Say, if you notice a rise in user frustration, you'll want to review specific problematic conversations. You can export examples to fine-tune the model or create a test set to adjust your prompts to fix the issue.

Also, once your LLM judge is live, it's important to check in on it regularly. Even an aligned LLM evaluator — or your expectations — can shift over time, so you might need to adjust or add new judges to keep things running smoothly.

LLM Observability. You could code this workflow step by step — setting up app instrumentation, log storage, job orchestration, dashboards, and alerts. Purpose-built LLM observability tools like **Evidently Cloud** streamline the process. You can **sign up for free** and get started quickly without the need to spin up all the infrastructure yourself.

Pros and Cons

TL;DR: Pros are reasonable quality, speed, ease of updates, and the ability to run evals for custom criteria without a reference. Cons are imperfect quality, potential biases, and the costs and risks of using external LLMs.

LLM judges offer a practical, scalable solution for evaluating AI outputs, but they come with trade-offs. Here's a breakdown of its pros and cons.

Pros of LLM judges

High-quality evaluations. When properly set up, LLM evaluations closely match human judgment and provide consistent results. For large-scale tasks, they're often the only viable alternative to human review.

No reference is needed. You can use LLMs to evaluate generated outputs without a reference answer to compare to. This makes them perfect for live monitoring in production.

It's incredibly flexible. You can tweak prompts to evaluate anything, from how helpful a response is to how well it matches a brand's voice, much like assigning tasks to humans.

It's scalable. Once configured, LLMs can handle thousands of evaluations around the clock, much faster and cheaper than human reviewers. Their fast turnaround and real-time availability are hard to beat. Plus, you can easily scale evaluations across multiple languages.

It's easy to update. LLM evaluators are simple to adjust as your needs and criteria change together with your product. Just update the prompt — no need to retrain a model, like in classic machine learning.

You can involve domain experts. Since LLMs use natural language, even non-technical team members can help write prompts and review results, ensuring the evaluations capture the right nuances.

Cons of LLM judges

Good, but not perfect. LLM evaluators aren't flawless. If the instructions are too vague or complex, the results can be subpar. LLMs might also give inconsistent judgments for the same input, though polling (asking the same question multiple times) can help. It's important to manage expectations and track quality.

Bias risks. LLMs are trained on vast datasets, which can carry biases into evaluations. For example, if you ask it to classify texts as "professional" or "unprofessional" without guidelines, it will rely on assumptions from its training data. For pairwise evals, there are known biases:

- **Position bias:** Favoring responses based on their placement (e.g., first or last).
- **Verbosity bias:** Preferring longer answers, even if they aren't more accurate.
- **Self-enhancement bias:** Favoring texts generated by the same LLM.

You can manage these by switching the order of responses or focusing on direct scoring rather than asking the LLM to pick the "best" response.

Research: These biases were discussed in the paper *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena* (Zheng, L., 2023).

Data privacy. Using third-party LLM APIs for evaluations means sharing data externally. While this is often fine, considering you're already generating responses through LLMs, you need to be cautious with privacy and security — especially for sensitive data.

It's not instant. While LLMs are faster than humans, they're still slower and more expensive compared to rule-based checks or smaller ML models. This makes them less ideal for tasks like guardrails, where you need to validate an answer right as it's being generated.

It's not free. Running evaluations with powerful LLMs can get expensive, especially with large datasets or multiple calls per generation. You can cut costs by mixing LLM judgments with simpler methods like regex, using sampling, and using more efficient evaluation prompts — running a single call rather than in multiple steps.

It takes work to set up. Building an effective LLM judge requires effort. You'll need to define criteria, design evaluation prompts, and label datasets. It's not a "set-it-and-forget-it" solution either — you'll need to check in now and then to keep your judge performing well.

LLM judge alternatives

TL;DR: Other options include manual labeling, collecting user feedback, using task-specific ML models and rule-based checks. When a reference is available, you can use metrics like ROUGE. A mix of methods often works best.

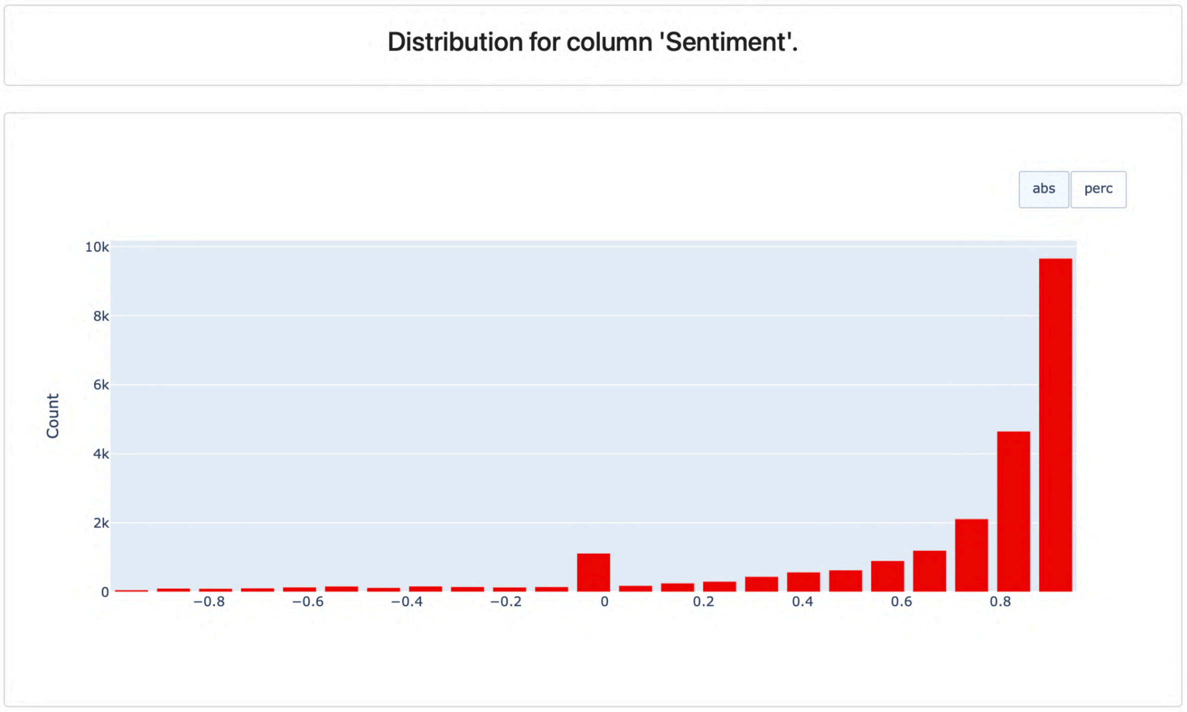
While LLM judges are great, they're not the only way to evaluate AI outputs. Here are some other LLM evaluation methods to consider.

Manual labeling. Human evaluation is still the gold standard, especially for tasks that are more nuanced or subjective. It doesn't scale for large volumes, but starting with manual reviews almost always makes sense. This helps you figure out the behaviors and patterns you want to track. Once you've nailed down your evaluation guidelines, you can automate them with LLMs. It's always a good idea to keep reviewing a portion of texts manually.

Collect user feedback. Don't forget your users! You can ask them to rate the quality of an LLM's response directly in-app, like right after it generates a response. This gives you real-time feedback from the people actually using the system.

Purpose-built ML models. For well-defined tasks, traditional machine learning models can be very effective. There are plenty of **pre-trained models** available for things like sentiment analysis or detecting personal information (PII). You can also use embedding models to compare **semantic similarity** between texts, which is useful for things like regression testing when you need to compare old and new responses.

Fine-tuned evaluator models. As you use LLM judges, you'll naturally gather a labeled dataset that you can use to fine-tune models for specific evaluation tasks. This can be a great long-term solution if your tasks are predictable, but these models have limitations. They're usually locked into their specific evaluation schemes and won't generalize as well if the inputs or criteria change (see paper).



For example, you can evaluate sentiment of texts using readily available ML models.

Generative quality metrics. For offline evaluation of structured tasks like summarization and translation, you can use metrics like ROUGE and BLEU to objectively measure response quality by comparing them to reference texts. These metrics use n-gram overlap to score how closely the output matches an ideal response.

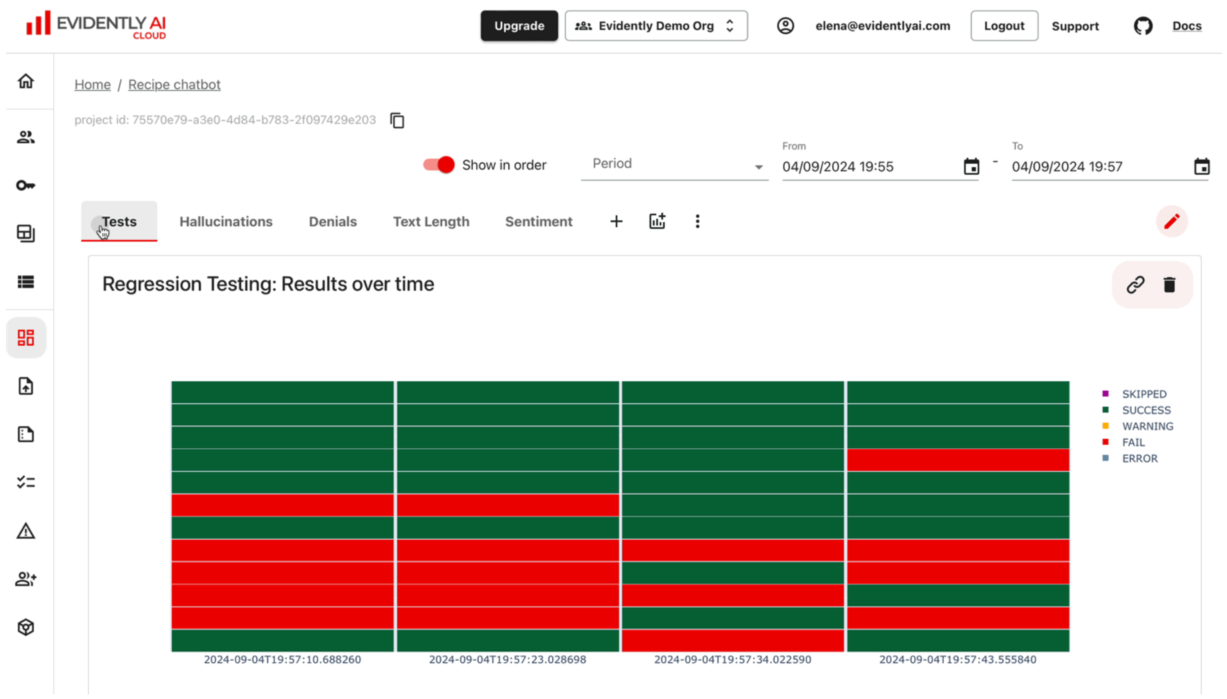
Rule-based checks. Sometimes the simplest methods work best. Using rules, like searching for specific keywords or phrases (e.g., "cannot" or “apologies” for denial, or a list of profane words), helps catch obvious errors quickly. It’s fast, cheap, and straightforward, plus it’s deterministic — you know exactly what you’re getting.

In practice, you combine methods. For instance, you could start with rule-based systems to filter out obvious issues, and then use LLM judges or human reviews for more complex, nuanced tasks.

Create an LLM judge for your AI system

LLM-as-a-Judge is a scalable alternative to human labeling. If you are working on complex AI systems like RAG or AI agents, you'll need such task-specific evaluations tuned to your criteria and preferences. That's why we built Evidently, an open-source framework with over 25 mn downloads. It makes creating and running LLM-as-a-Judge evaluations easy.

On top of it, Evidently Cloud provides a platform to collaboratively run tests and evaluations. You can generate synthetic test data, create LLM judges with no-code, and track performance over time — all in one place.



Ready to create your first LLM judge?
Schedule a demo to see Evidently Cloud in action. We're here to help you build AI systems with confidence!